DESARROLLO DE UNA APLICACIÓN MIXTA MÓVIL/WEB PARA EL SEGUIMIENTO DE DIETAS

DEVELOPMENT OF A MIXED MOBILE/WEB APPLICATION FOR DIETARY MONITORING

Personal Diet

Evald Alin Artene - Grado en Ingeniería Informática

Ignacio Baena Kuroda - Grado en Ingeniería de Software

FACULTAD DE INFORMÁTICA

Universidad Complutense de Madrid

TRABAJO DE FIN DE GRADO
CURSO 2020-2021

DIRECTOR
Antonio Sarasa Cabezuelo

Agradecimientos

En primer lugar, se quiere dar las gracias al tutor del proyecto, Antonio Sarasa Cabezuelo, por la paciencia y por guiarnos a través del proyecto.

En segundo lugar, agradecer también al experto en nutrición, Juan Francisco Mirabet Delgado, que nos ha asesorado en los primeros compases de la aplicación sobre lo que él, como profesional del sector, querría en una aplicación de este calibre.

A los compañeros de universidad, amigos, que nos han ayudado a nivel usuarios de la aplicación, y a nivel personal, para realizar el proyecto.

A Antonio Prieto, nuestro jefe en el ámbito laboral, que nos ha apoyado y empujado para la finalización del proyecto, así como el grado.

A Aldana Subero, por aportar el diseño original y realización de las imágenes que aparecen en la aplicación móvil.

Y, por último, y más importante, a nuestras familias y parejas, que nos han brindado un constante apoyo y empuje, sobre todo a nivel anímico, que después de estos años de universidad, y sobre todo en los últimos empujones, es cuando más se ha necesitado.

Resumen

En esta memoria se recoge la especificación, diseño e implementación de una aplicación que combina una parte web con una parte de aplicación móvil, más concretamente, para Android. Se recogen los componentes del sistema, los servicios que se usan en la aplicación, así como imágenes y figuras sobre el desarrollo y acabado del sistema.

En este trabajo se presenta una aplicación orientada a la ayuda de ciertos usuarios, los llamados pacientes, a llevar un control más exhaustivo de lo que ingieren con ayuda de otros usuarios, los dietistas, con el objetivo de mejorar su salud y condición física a través de la correcta ingesta de alimentos.

El paciente, que usa el sistema mediante una aplicación móvil, debe elegir al dietista por el que quiere ser aconsejado (aunque este se puede reservar el derecho de aceptarles o no), con el que tendrán un intercambio continuo de información sobre la dieta y su seguimiento a través de la aplicación móvil, y con el contacto que ellos prefieran tener, que será número de teléfono o correo.

El dietista, que usa el sistema mediante una aplicación web, acepta los pacientes que quiere tener a su cargo, les crea una dieta personalizada, la puede modificar, y puede seguir el progreso de estos con la dieta que les ha asignado.

El administrador, que también usará el sistema mediante la aplicación web, puede aceptar o rechazar a los dietistas que se registren en la aplicación, así como borrar usuarios, tanto pacientes como dietistas, mediante una sencilla interfaz.

PALABRAS CLAVE:

aplicación Android, aplicación web, dieta, nutrición, hábitos alimenticios saludables, progreso físico, progreso personal

Abstract

This report includes the specification, design and implementation of an application that combines a web part with a mobile application part, more specifically, for Android. The components of the system, the services used in the application, as well as images and figures about the development and finishing of the system are collected.

This paper presents an application aimed at helping certain users, the so-called patients, to carry out a more exhaustive control of what they ingest with the help of other users, dietitians, with the aim of improving their health and physical condition through the correct intake of food.

The patient, who uses the system through a mobile application, must choose the dietitian for whom he wants to be advised (although this can be reserved the right to accept them or not), with whom they will have a continuous exchange of information about the diet and its follow-up through the mobile application, and with the contact that they prefer to have, which will be phone number or mail.

The dietitian, who uses the system through a web application, accepts the patients he wants to take care of, creates a personalized diet, can modify it, and can track their progress with the diet he has assigned them.

The administrator, who will also use the system through the web application, can accept or reject dietitians who register in the application, as well as delete users, both patients and dietitians, through a simple interface.

KEYWORDS:

Android application, web application, diet, nutrition, healthy eating habits, physical progress, personal progress

Índice

Contenido

Índice de figuras

Capítulo 1: Introducción

En un mundo en el que lo común es tener un constante bombardeo de información, en ocasiones de información tergiversada y falaz, que distorsiona la actitud frente a la comida. Muchas personas están en guerra con los alimentos, cada uno con un motivo, pero que no debería ser así, se debería tener una relación conciliadora con la comida y ser consciente en todo momento de los pros y contras que tiene cada alimento.

Con esta idea de conciliación, y con el afán por aprender tecnologías en las que en el grado no se han podido profundizar, sale el desarrollo de este sistema de gestión de dietas alimenticias, que pone en contacto a profesionales de la nutrición y gente que quiere aprender a comer, en una aplicación mixta web/Android.

Con este proyecto se pretende cubrir la necesidad de las personas que desean mejorar sus hábitos alimenticios, como el desarrollo de un entorno laboral, y un posible nicho de mercado a los profesionales de la salud que quieran comercializar, en primera instancia, con el conocimiento que ellos tienen por experiencia que puedan adquirir.

1.1 MOTIVACIÓN

Siempre se está en un debate continuo, con un dilema que, a veces y de manera equivocada, se torna en un debate moral con uno mismo acerca de los alimentos que se toman. Se vive en una sociedad en la que se ha convertido lo que no es tan bueno para la salud en alimentos que se consumen diariamente, a sabiendas de que no es lo más correcto. A todos nos gusta disfrutar de ultra-procesados como la bollería o ciertos alimentos preparados, o de un vino o una cerveza después de la comida, pero no se debe caer en la trampa de convertir esto en algo cotidiano que se piensa que nunca pasará factura.

La gran mayoría de personas han pensado, al menos una vez en la vida, que deberían mejorar su estilo de vida, que no tienen unos hábitos alimenticios saludables y que algo deben hacer para cambiar esto. A esa gran mayoría también les ha surgido el problema de no saber cómo llevar a cabo ese cambio.

Con estas premisas, surge la idea de crear una aplicación que ayude a esas personas a través de una conexión directa con expertos en nutrición.

1.2 OBJETIVOS

El objetivo principal de este Trabajo de Fin de Grado es la creación de un sistema que mezcla una aplicación web y una aplicación Android que permite, por una parte, a unos

usuarios que quieren aprender a comer mejor, a conseguirlo; y, por otra parte, a los expertos en nutrición en ayudar a conseguir ese objetivo a las personas con un afán vocacional, así como un posible interés laboral.

Los objetivos específicos que plantea el proyecto son los siguientes:

- Realizar una aplicación web que sea capaz de proporcionar a los expertos en nutrición, los dietistas, las herramientas adecuadas para la creación de planes de dietas para los usuarios que tiene a cargo.
- Realizar la aplicación web con una interfaz amigable y simple para el/los administradores/es del sistema con el objetivo de ser capaz de mantener, a nivel de usuarios, el sistema de una manera muy sencilla.
- Mostrar en la aplicación web, concretamente a los dietistas, los progresos que van manteniendo los usuarios que tienen asignados.
- Diseñar una aplicación móvil que sea muy intuitiva como para que cualquier usuario sea capaz de registrar se y actualizar su información personal y de contacto.
- Diseñar la aplicación móvil de tal manera en que los usuarios sean capaces de tener un seguimiento rápido de la dieta que deben seguir, así como actualizar el progreso personal y sobre la dieta.
- Establecer alertas en la aplicación móvil para que los usuarios sean capaces de seguir la dieta.
- Tener un registro de los progresos de los pacientes para con su dietista y dieta.
- Control sobre el acceso en la aplicación web, de tal manera que cualquier persona puede registrarte, pero deben ser aceptados previamente por los administradores del sistema.
- Tener un sistema de valoración en los dietistas, de tal manera que los usuarios pueden valorar su trabajo y eso repercute en las posibles y futuras elecciones de otros usuarios sobre los servicios de los dietistas.

1.3 ESTADO DEL ARTE

En el mercado existen numerosas aplicaciones enfocadas a dar un servicio que tenga que ver con la alimentación y nutrición de las personas. Hay aplicaciones que permiten monitorizar la comida que se ingiere teniendo un seguimiento de datos como son: las calorías de dichas comidas, sus macronutrientes, la cantidad de estas, el momento en el que se toman, etc. En esta línea, se tienen los ejemplos de "***My FitnessPal"*** o ***"FatSecret".*** Hay otras aplicaciones que están enfocadas en poner en contacto expertos en condición y salud física, como entrenadores personales y nutricionistas, con gente que quiera dichos servicios. En esta otra línea, lo más común es encontrar páginas web estilo Blog de estos expertos, hoy en día la mayoría conocidos a través de las redes sociales.

Personal Diet pretende romper con la separación de servicios mencionada arriba, pues el objetivo es poner en contacto a estas personas que quieren prestar servicios de nutrición con gente que los quiera disponer. También el enfoque que se le da a ***Personal Diet*** es

distinto a las aplicaciones y/o webs mencionadas arriba, pues se pretende que los expertos que brinden servicios usen la web como una herramienta de trabajo, pues tiene una interfaz sencilla de seguimiento de usuarios, así como del seguimiento de sus dietas. Por otra parte, el diseño de la aplicación móvil, la que es usada por los pacientes usuarios de ***Personal Diet,*** tiene un enfoque mucho más atractivo para un usuario estándar, pues no se pretende que estos usuarios usen la aplicación de un modo corporativo, si no tomarlo como algo mucho más interactivo, de tal modo que esto favorezca el uso de la aplicación por parte de estos usuarios, y que no la acaben desechando.

A continuación, se presentan algunas aplicaciones y/o webs personales que tienen similitudes con este proyecto:

- ***My FitnessPal:*** esta es una de las aplicaciones más ampliamente usadas en el mercado. Se trata de un llamado "Contador de calorías", pues la esencia de esta aplicación es anotar los alimentos que se han ingerido en un determinado momento con sus cantidades, y la aplicación guarda un registro de esto, mostrando datos como son las calorías ingeridas, las proteínas que tenía, etc. También se pueden establecer ciertos datos de información personal, como son el peso o el ICM, con el objetivo de ir comprobando, por cuenta de uno mismo, estos datos, y actualizarlos si fuera pertinente, para así ver el progreso que se est teniendo con las comidas está siendo fructífero o no.
- ***FatSecret:*** Sus funcionalidades son muy parecidas a la aplicación del punto anterior, en el que se difiere, en esencia, en la interfaz que se presenta.
- ***Webs personales:*** En este ámbito, lo más común para encontrar personas que te puedan guiar en el objetivo de tener una mejor alimentación, son webs personales estilo Blog de cada nutricionista en particular. Lo más común en estas webs es enseñar información de contacto sobre el nutricionista en cuestión, así como los planes de dietas que se pueden elegir (por un precio), e imágenes sobre otros clientes de estos, mostrando los avances conseguidos con ellos.

1.4 ESTRUCTURA DE LA MEMORIA

La memoria está organizada según los estándares que están propuestos para la realización de proyectos de este calibre. La memoria consta de diez capítulos, un apartado con agradecimientos, otro apartado con el prólogo del TFG, un resumen (con su debida traducción en inglés), dos índices, uno de los capítulos y otro de las figuras empleadas, y por último, la bibliografía empleada.

A continuación, se presenta una pequeña descripción sobre cada uno de los capítulos:

1. **Introducción**

   En este capítulo, se presenta un pequeño ápice sobre lo que es el proyecto en sí. Se explica de donde y por qué sale la idea, así como lo que se pretende conseguir con el TFG. A continuación, se detalla la existencia parcial de aplicaciones y servicios en el mercado que pretenden cubrir las mismas necesidades que este proyecto. Acabando, se presenta la estructura que tiene la presente memoria.

2. Especificación de la aplicación

En este capítulo se detalla el alcance y funcionalidades de la aplicación móvil, así como de la aplicación web. Para ello, se estructura el capítulo en módulos pertinentes a los actores de la aplicación, que se acompañan de diagramas para los casos de uso que componen la aplicación.

3. Tecnología empleada

En este capítulo se detalla el alcance y funcionalidades de la aplicación móvil, así como de la aplicación web. Para ello, se estructura el capítulo en módulos pertinentes a los actores de la aplicación, que se acompañan de diagramas para los casos de uso que componen la aplicación.

4. Arquitectura de la aplicación

En este capítulo se detalla de una manera esquemática como se comunican las dos aplicaciones entre sí y como se consigue y mediante qué mecanismos, el intercambio y procesamiento de información.

5. Modelo de datos

En este capítulo se detalla cómo es la estructura de la información que se alberga en la base de datos, y de las distintas colecciones que forman parte de esta, así como la explicación de cada una de ellas de manera más detallada.

6. Diseño

En este capítulo se detalla el diseño que se ha seguido para realizar la aplicación móvil y la aplicación web. Se detallan el por qué de ciertos aspectos de las aplicaciones, como el por qué de la sencillez propuesta en ambas aplicaciones.

7. Implementación

En este capítulo se detalla cómo se ha desarrollado la aplicación web a nivel de código, como se han implementado los diferentes casos de uso ejemplificando esto con figuras que tienen fragmentos de código o archivos usados en el desarrollo e implementación del proyecto.

8. Conclusiones y trabajo futuro

En este capítulo se detallan las conclusiones y consecución de objetivos del proyecto, así como la propuesta de llevar el proyecto a una línea en la que se tenga

un despliegue para todo usuario que quisiera usar las aplicaciones y las posibles mejoras y ampliaciones futuras de las aplicaciones.

9. Aportaciones individuales

En este capítulo se detalla el trabajo realizado por los dos integrantes que han desarrollado el proyecto, así como los conocimientos y tecnologías adquiridas por cada uno.

Chapter 1: Introduction

1.1 MOTIVATION

In a world in which it is common to have a constant bombardment of information, sometimes of distorted and fallacious information, which distorts our attitude towards food. Many people are at war with food, each with a motive, but that should not be the case, we should have a conciliatory relationship with food and be aware at all times of the pros and cons that each food has.

With this idea of reconciliation, and with our eagerness to learn technologies in which in the degree we have not been able to get, comes the development of this food management system, which puts together nutrition professionals and people who want to learn to eat, in a mixed web / Android application.

This project aims to cover the need of people who want to improve their eating habits, such as the development of a work environment, and a possible market niche for health professionals who want to market, in the first instance, with the knowledge they have from experience that they can acquire.

1.2 OBJETIVES

The main objective of this End of Degree Work is the creation of a system that mixes a web application and an Android application that allows, on the one hand, some users who want to learn to eat better, to get it; and, on the other hand, nutrition experts in helping people achieve that goal with a vocational zeal as well as a possible job interest.

The specific objectives of the project are as follows:

- Make a web application that can provide nutrition experts, dietitians, with the right tools for creating diet plans for the users you oversee.
- Make the web application with a friendly and simple interface for the system administrator(s) to be able to maintain, at the user level, the system in a very simple way.
- Show in the web application, specifically to the dietitians, the progress that the users assigned to them are maintaining.
- Design a mobile application that is too intuitive for any user to be able to register and update their personal and contact information.

- Design the mobile app in such a way that users can have a quick follow-up of the diet they should follow, as well as update personal and dietary progress.
- Set alerts in the mobile app so that users can follow the diet.
- Keep track of patients' progress towards their dietitian and diet.
- Control over access in the web application, so that anyone can register, but they must be accepted in advance by the system administrators.
- Have a rating system for the dietitians, so that users can value their work and that affects the possible and future choices of other users about dietitian's services.

1.3 STATE OF THE ART

In the market there are numerous applications focused on providing a service that has to do with the food and nutrition of people. There are applications that allow you to monitor the food that is ingested by keeping track of data such as: the calories of these meals, their macronutrients, the amount of these, the time in which they are taken, etc. In this line, we have the examples of ***"My FitnessPal" or*** "***FatSecret".*** There are other applications that are focused on putting experts in fitness and physical health, such as personal trainers and nutritionists, in contact with people who want such services. In this other line, the most common thing is to find blog-style web pages of these experts, nowadays most known through social networks.

Personal Diet aims to break with the separation of services mentioned above, because the aim is to connect these people who want to provide nutrition services with people who want to have them. Also, the approach that is given to Personal ***Diet*** is different from the applications and / or websites mentioned above, since it is intended for experts who want to provide services and use the web as a work tool, since it has a simple interface for tracking users, as well as monitoring their diets. On the other hand, the design of the mobile application, which is used by patient users of Personal ***Diet,*** has a much more attractive approach for a standard user, since it is not intended that these users use the application in a corporate way, but take it as something much more interactive, in such a way that this favors the use of the application by these users, and that they do not end up discarding it.

Below are some applications and/or personal websites that have similarities with this project:

- ***My FitnessPal:*** this is one of the most widely used applications on the market. It is a so-called "Calorie Counter", because the essence of this application is to write down the foods that have been ingested at a certain time with their amounts, and the application keeps a record of this, showing data such as the calories ingested, the proteins you had, etc. You can also establish certain personal information data, such as the weight, in

order to check on your own account, this data, and update it if relevant, in order to see the progress we are having with the meals we are eating.

- ***FatSecret:*** Its functionalities are very similar to the application of the previous point, in which it differs, in essence, in the interface that is presented.
- ***Personal websites:*** In this area, the most common to find people who can guide you in the goal of having a better diet, are personal blog-style websites of each nutritionist. The most common thing on these websites is to show contact information about the nutritionist in question, as well as the diet plans that can be chosen (for a price), and images about other clients of these, showing the progress made with them.

1.4 MEMORY STRUCTURE

The memory is organized according to the standards that are proposed for the realization of projects of this caliber. The report consists of ten chapters, a section with acknowledgements, another section with the prologue of the TFG, a summary (with its proper translation in English), two indexes, one of the chapters and another of the figures used, and finally, the bibliography used.

Below is a brief description of each of the chapters:

1. **Introduction**

   In this chapter, a small iota is presented about what the project itself is. It explains where and why the idea comes from, as well as what it is intended to achieve with the TFG. Below, the partial existence of applications and services in the market that aim to cover the same needs as this project is detailed. Finishing, the structure of the present memory is presented.

2. **Specifying the application**

   This chapter details the scope and features of the mobile application as well as the web application. To do this, the chapter is structured in modules relevant to the actors of the application, which are accompanied by diagrams for the use cases that make up the application.

3. **Technology used**

   This chapter details the scope and features of the mobile application as well as the web application. To do this, the chapter is structured in modules relevant to the

actors of the application, which are accompanied by diagrams for the use cases that make up the application.

4. Application architecture

This chapter details in a schematic way how the two applications communicate with each other and how it is achieved and through what mechanisms, the exchange and processing of information.

5. Data model

This chapter details the structure of the information contained in the database, and the different collections that are part of it, as well as the explanation of each of them in more detail.

6. Design

This chapter details the design that has been followed to make the mobile application and the web application. The reason for certain aspects of the applications is detailed, such as the simplicity proposed in both applications.

7. Implementation

This chapter details how the web application has been developed at the code level, how the different use cases have been implemented exemplifying this with figures that have code snippets or files used in the development and implementation of the project.

8. Conclusions and future work

This chapter details the conclusions and achievement of objectives of the project, as well as the proposal to take the project to a line in which it has a deployment for any user who would like to use the applications and the possible improvements and future expansions of the applications.

9. Individual contributions

This chapter details the work done by the two members who have developed the project, as well as the knowledge and technologies acquired by each.

Capítulo 2: Especificación de la aplicación

En este capítulo se van a especificar los requisitos de la aplicación. Para ello, en primer lugar, se describen los actores que interactúan con la aplicación, y a continuación se desarrollan los casos de uso en los que intervienen los actores agrupados por módulos funcionales.

2.1 ACTORES

En la aplicación de este trabajo se han definido los siguientes actores:

- Paciente: es el usuario que desee usar la aplicación para mejorar su dieta, y es el que use la mayor parte de las funcionalidades de la aplicación.
- Dietista: es el usuario que realiza el mantenimiento de las dietas que aparecen en la aplicación de los pacientes que tiene asignado.
- Administrador: es el encargado de mantener la aplicación.

Las funcionalidades de la aplicación se han agrupado en los siguientes módulos funcionales: módulo usuario, módulo paciente y módulo dietista.

2.2 MÓDULO USUARIO

En este módulo, se encuentran los casos de uso que engloban las funcionalidades básicas de cualquier aplicación que tenga usuarios en sus bases de datos, es decir, funcionalidades como el registro en la aplicación, iniciar sesión, ver y modificar datos.

Ilustración 1: Diagrama de casos de uso Usuarios

01	Darse de alta
Actor	Dietista, Paciente, Administrador
Descripción	El usuario se da de alta en la aplicación ya sea como Dietista (Web) o Paciente (Android)
Precondición	El usuario no debe de estar dado de alta previamente
Secuencia normal	1. El sistema muestra un botón <<Registrarse>> 2. En el caso del paciente, el sistema presenta el siguiente formulario: 1. Email

	2. Contraseña 3. Repetir contraseña 3. En el caso del dietista, el sistema presenta el siguiente formulario: a. Nombre y apellidos del usuario b. Contraseña c. Repetir contraseña d. Email e. Teléfono f. Descripción dietista
Postcondición	El sistema genera automáticamente un registro con los datos El sistema informa al usuario que se ha dado de alta
Excepciones	Se introduzcan los campos del formulario con formatos inválidos, lo que hace que salte un error notificando de esto al usuario.
Comentarios	El administrador puede dar de alta a cualquier tipo de usuario en una interfaz similar al rol que desee añadir.

02	Iniciar Sesión
Actor	Paciente, Dietista, Administrador
Descripción	El usuario puede iniciar sesión en la aplicación y navegar en ella.
Precondición	El usuario debe de estar dado de en la aplicación
Secuencia normal	1. El usuario introduce su nombre de usuario y su contraseña en un formulario. 2. Pulsa en “Validar” y entra en la aplicación si los datos son correctos.
Postcondición	A. Si los datos son correctos entra en la aplicación B. Si los datos son incorrectos se despliega un mensaje donde se informa del error.

Excepciones	Si el paciente es la primera vez que inicia sesión se redirige al usuario a una ventana donde se debe introducir todos los siguientes datos: 1. Nombre y Apellidos 2. Teléfono 3. Edad 4. Peso 5. Altura 6. Sexo 7. Frecuencia de actividad deportiva Una vez introducidos todos los campos el usuario pulsará en <<Confirmar>>
Comentarios	

03	Cerrar Sesión
Actor	Paciente, Dietista, Administrador
Descripción	El usuario puede cerrar sesión
Precondición	El usuario debe haber iniciado sesión previamente.
Secuencia normal	1. El usuario puede pulsar en “Cerrar sesión” en un desplegable en la ventana en la pantalla de inicio. 2. Se despliega un pop up donde se pide confirmación de que el usuario realmente quiere cerrar sesión.
Postcondición	El sistema cierra la sesión del usuario.
Excepciones	
Comentarios	

04	Ver perfil
Actor	Paciente, Dietista, Administrador
Descripción	El usuario puede ver sus datos personales

Precondición	El usuario debe estar dado de alta en la base de datos. El usuario debe de estar logueado en la aplicación
Secuencia normal	Al iniciar sesión el paciente ve una ventana con la siguiente información. 1. Imagen de perfil 2. Peso 3. IMC (Índice de Masa Corporal)
Postcondición	El sistema termina mostrando información relacionada con el usuario.
Excepciones	
Comentarios	La información que se muestra cambia en función de si se es un dietista o un paciente. El administrador puede ver el perfil de otros usuarios.

05	Modificar usuario
Actor	Paciente, Administrador
Descripción	El paciente puede editar sus datos
Precondición	El paciente debe de estar logueado en la aplicación
Secuencia normal	1. El paciente pulsa en icono flotante el menú de perfil 2. Se despliega una nueva ventana donde podrá modificar los siguientes campos 1. Nombre y Apellidos 2. Teléfono 3. Edad 4. Altura 5. Foto de perfil 3. El paciente pulsa en <<CONFIRMAR>> con los nuevos datos deseados.
Postcondición	El sistema actualiza al usuario en la Base de datos

Excepciones	
Comentarios	En el caso del paciente, si cambia de Altura, Peso, Frecuencia de Actividad y/o Fecha de Nacimiento recalcula el objetivo recomendado y avisa de ello. El administrador puede modificar ciertos datos de los dietistas, como son el nombre y apellidos, el teléfono y la descripción, mediante un breve formulario.

06	Darse de baja
Actor	Paciente, Dietista y Administrador
Descripción	El usuario se da de baja en la aplicación
Precondición	El usuario debe de estar dado de alta previamente
Secuencia normal	1. El sistema muestra un botón <<Dar de baja>> en el perfil del usuario en cuestión. 2. Hay una doble confirmación de esta acción, pues es un cambio irreversible 3. En doble caso afirmativo, se borra al usuario de la base de datos.
Postcondición	El sistema pone como inactivo al usuario. Si es un dietista, a todos los pacientes que tiene asociados se les notifica y estos pueden volver a elegir a otro dietista.
Excepciones	
Comentarios	El paciente y el dietista lo hacen de igual forma. El administrador puede acceder a los perfiles de todos los demás usuarios y darles de baja.

07	Ver todos usuarios
Actor	Administrador

Descripción	El Administrador puede acceder a una lista de todos los usuarios dados de alta en la aplicación.
Precondición	El usuario debe de estar dado de alta.
Secuencia normal	1-El administrador pulsa en <<Manejar pacientes>> o en <<Manejar Dietistas>> y el sistema despliega una lista con todos los usuarios en función del botón pulsado.
Postcondición	El sistema muestra todos los usuarios existentes en la aplicación.
Excepciones	
Comentarios	

08	Recuperar contraseña
Actor	Paciente, Dietista, Administrador
Descripción	El usuario puede recuperar su contraseña si la olvida.
Precondición	El usuario debe de estar dado de alta.
Secuencia normal	1. El usuario pulsa en <<He olvidado mi contraseña>> 2. Se abre una ventana donde el usuario debe introducir su email. 3. Pulsa en << Enviar>> a. Si se ha introducido un correo existente en la base de datos el paciente recibe un correo en su bandeja de entrada con un link para restablecer su contraseña b. En caso contrario muestra un mensaje de error
Postcondición	El sistema restablece la contraseña del usuario enviando un email
Excepciones	
Comentarios	En el caso del dietista, el botón se llama <<Reestablecer contraseña>>, y se le envía un correo en el que puede reestablecer la contraseña.

2.3 MÓDULO PACIENTE

En este módulo, se encuentran los casos de uso pertinentes al seguimiento de una dieta por parte de un paciente.

Las funcionalidades más abajo explicadas son las referentes al seguimiento de su progreso y la actualización de este, el seguimiento de la dieta de cada día, así como su actualización, el seguimiento de todo el progreso desde que el usuario entró en la aplicación, así como ver los datos de su dietista.

Además, paciente puede ver publicaciones que su dietista le asigne, puede conseguir logros y por tanto ver que logros ha obtenido.

También puede hacer uso de una guía sobre mejorar los hábitos alimenticios.

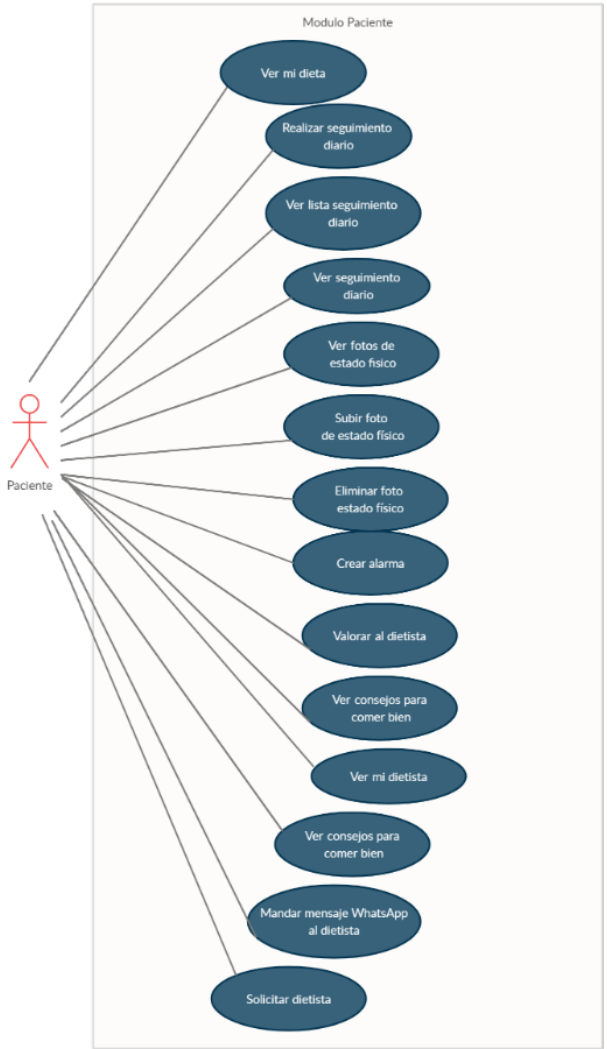


Ilustración 2:Diagrama de casos de uso Paciente

A continuación, se encuentran los casos de uso de este módulo.

09	Ver mi dieta
Actores	Paciente
Descripción	El paciente puede ver la información de la dieta que tiene asignada por el dietista.
Precondición	Debe existir el paciente, y que tenga una dieta asignada, así como un dietista.
Secuencia normal	1. El paciente se encuentra en su ventana de inicio. 2. Accede al apartado <<MI DIETA>> al pulsar el botón correspondiente. 3. Se muestra la información de la dieta separado en días de lunes a domingo. 4. Si el paciente pulsa en un día se despliega la información de la dieta acorde a ese día.
Postcondición	Se muestra la información de la dieta.
Excepciones	
Comentarios	Dentro de la información de la dieta se ven campos como la descripción de la dieta, de que está compuesta, objetivo de la dieta, etc. Esta información viene dada por el dietista

010	Realizar seguimiento diario
Actores	Paciente
Descripción	El paciente puede crear un registro reflejando si ha seguido las pautas de la dieta en el día actual.
Precondición	Debe existir el paciente, y que tenga una dieta asignada, así como un dietista. No debe haber realizado el seguimiento del día anteriormente
Secuencia normal	1. El paciente se encuentra en la ventana de inicio 2. El paciente pulsa en <<SEGUIMIENTO DE DIETA>>

	3. Se abre una ventana donde aparece un listado de todas las comidas o ayunos que debe hacer el paciente junto a una *checkbox* para indicar si la ha realizado o no 4. El paciente puede introducir una descripción de cómo fue su día. 5. Si pulsa en el botón con el icono de una cámara se abre la galería del móvil y puede elegir que fotos añadir a la lista. Las fotos que se añaden aparecen en la vista general. 6. Si pulsa en <<VALIDAR DATOS>> se abre una ventana emergente que espera una confirmación de la acción. 1. Si la respuesta es SI se creara el registro. 2. Si la respuesta es NO el paciente puede seguir editando el seguimiento.
Postcondición	Se crea un registro de seguimiento para el día actual.
Excepciones	
Comentarios	Solo se puede realizar una vez al día. No se puede modificar una vez se ha creado el registro.

011	Ver lista seguimiento diario
Actores	Paciente
Descripción	El paciente puede ver una lista con los seguimientos diarios de la dieta que está realizando.
Precondición	Debe existir el paciente, y que tenga una dieta asignada, así como un dietista.
Secuencia normal	1. El paciente se encuentra en la ventana <<MI DIETA>> 2. Accede al apartado <<MIS SEGUIMIENTOS>> al pulsar el botón correspondiente. 3. Aparece una nueva ventana con todos los seguimientos realizados hasta día de hoy.
Postcondición	Se muestra la lista del seguimiento diario del paciente.
Excepciones	

Comentarios	La información por mostrar es la relevante a las comidas que debe realizar o que ha realizado el paciente en el día.

012	Ver seguimiento diario
Actores	Paciente
Descripción	El paciente puede ver los seguimientos diarios de la dieta que está realizando.
Precondición	Debe existir el paciente, y que tenga una dieta asignada, así como un dietista.
Secuencia normal	1. El paciente se encuentra en la ventana << MIS SEGUIMIENTOS >> 2. Pulsa en el día que desea ver. 3. Aparece una ventana de cómo se realizó el seguimiento por parte del paciente además de su peso ese día.
Postcondición	Se muestra el seguimiento diario del paciente.
Excepciones	
Comentarios	La información por mostrar es la relevante a las comidas que debe realizar o que ha realizado el paciente en el día.

013	Ver fotos de estado físico
Actores	Paciente
Descripción	El paciente puede ver una lista de fotos de su estado físico que están subidas.
Precondición	Debe existir el paciente Se deben tener permisos de cámara en la aplicación.
Secuencia normal	1. El paciente se encuentra en su ventana de inicio. 2. En el menú desplegable pulsa en <<Mis fotos>> 3. Aparece una lista con todas las fotos que el paciente tiene subidas

Postcondición	Muestra las imágenes subidas del paciente
Excepciones	
Comentarios	

014	Subir foto estado físico
Actores	Paciente
Descripción	El paciente puede subir una foto de su estado físico para que tanto él como el dietista puedan tener más feedback a la hora de ver progresos.
Precondición	Debe existir el paciente, y que tenga una dieta asignada, así como un dietista. Se deben tener permisos de cámara en la aplicación.
Secuencia normal	1. El paciente se encuentra en la venta de <<Mis fotos>> 2. Pulsa en << Subir foto>> 3. Aparece una ventana donde pulsando en <<Seleccionar foto>> se despliega el explorador de imágenes del teléfono móvil. 4. El paciente selecciona la foto que desea subir. 5. Pulsa en <<Subir foto>> 6. Aparece un mensaje de confirmación al elegir la foto
Postcondición	Se añade la foto a la lista de <<Mis fotos>>
Excepciones	No pueda subir la imagen, con lo que sale un error y se le pide al usuario que vuelva a intentarlo.
Comentarios	

015	Eliminar foto estado físico
Actores	Paciente
Descripción	El paciente puede eliminar la foto que quiera de su lista de fotos de estado físico
Precondición	Debe existir el paciente.

	Debe existir, al menos, una foto de progreso.

Secuencia normal	1. El paciente se encuentra en su ventana de <<Mis fotos>>
	2. El paciente mantiene el dedo sobre la foto que desea eliminar
	3. Se despliega una ventana emergente preguntando si desea realmente eliminar esa imagen.
	    1. En caso afirmativo la foto se eliminará y se redirigirá al menú principal.
	    2. En caso negativo se cerrará la ventana emergente.

Postcondición	Se eliminará la foto de la lista de fotos de estado físico del paciente.
Excepciones	
Comentarios	

016	Crear alarma
Actores	Paciente
Descripción	El paciente puede crear una alarma para que recuerde la hora a la que tiene que pesarse.
Precondición	El paciente debe iniciar sesión
Secuencia normal	1. El paciente se encuentra en su ventana de inicio 2. Accede al apartado <<Recordatorio>> al pulsar sobre el botón correspondiente. 3. Se abre una ventana en la que se puede elegir una hora del día. 4. Pulsa en <<Establecer Alarma>> y es redirigido al menú de alarmas del sistema Android donde puede ver una nueva alarma con el mensaje “ES HORA DE PESARTE!”
Postcondición	Se redirige al apartado de alarmas del sistema operativo de Android
Excepciones	Si el usuario pulsa el botón sin introducir una hora se muestra un mensaje de aviso.
Comentarios	

017	Ver mi dietista
Actores	Paciente
Descripción	El paciente puede ver los datos del dietista que tiene asignado.
Precondición	Debe existir el paciente, y que tenga una dieta asignada, así como un dietista. El dietista debe existir y tener al paciente asociado.
Secuencia normal	1. El paciente se encuentra en su ventana de inicio 2. Accede al apartado <<Mi dieta>> al pulsar sobre el botón correspondiente. 3. Se muestra la información de la dieta, al igual que un botón <<Mi dietista>>, que, al pulsar sobre él, lleva al paciente a otra ventana. 4. En esta ventana, el paciente puede ver los siguientes datos del dietista. 1. Nombre 2. Descripción 3. Teléfono de contacto 4. Email de contacto
Postcondición	Se muestran los datos pertinentes del dietista a su paciente asociado.
Excepciones	
Comentarios	

018	Mandar mensaje *WhatsApp* al dietista
Actor	Paciente
Descripción	El paciente puede abrir la aplicación de WhatsApp para comunicarse con el paciente
Precondición	El usuario debe de estar dado de alta y tener una dieta asignada.
Secuencia normal	1- El paciente se encuentra en la pestaña <<MI DIESTISTA>> 2- Pulsa el botón flotante con el icono de *WhatsApp* 3- Redirige a la aplicación *WhatsApp* abriendo una conversación con el número de teléfono de Dietista.

Postcondición	El Paciente ver la interfaz de *WhatsApp* con una conversación con el dietista.
Excepciones	
Comentarios	

019	Mandar email al dietista
Actor	Paciente
Descripción	El paciente puede enviar un email al dietista.
Precondición	El paciente debe de estar dado de alta y tener una dieta asignada.
Secuencia normal	1- El paciente se encuentra en la pestaña <<MI DIESTISTA>> 2- Pulsa encima de la dirección de correo del dietista. 3- Redirige a la aplicación de envió de email que decida el paciente
Postcondición	El Paciente ve la interfaz de la aplicación de Email que se decida.
Excepciones	
Comentarios	

020	Valorar al dietista
Actor	Paciente.
Descripción	El Paciente puede valorar al dietista de 0 a 5
Precondición	El usuario debe de estar dado de alta y tener una dieta asignada.
Secuencia normal	1- El paciente se encuentra en la pestaña <<MI DIESTISTA>> 2- El paciente selecciona la valoración del dietista con un número de estrellas. Cada estrella equivale a 1 y tiene la capacidad de dar media estrella por lo que valdría 0.5 3- Pulsa en <<ENVIAR VALORACIÓN>>

Postcondición	La valoración del dietista se recalcula con el nuevo valor asignado.
Excepciones	
Comentarios	

021	Ver consejos para comer bien
Actores	Paciente
Descripción	El paciente puede ver una guía referida a los buenos hábitos alimenticios
Precondición	Debe existir una guía de alimentación.
Secuencia normal	1. Desde el menú de inicio puede acceder desde el botón <<Consejos para comer bien>> 2. A continuación, se muestra una serie de imágenes referidas los buenos hábitos alimenticios. 3. Se puede pasar de imagen en imagen deslizando el dedo de izquierda a derecha
Postcondición	Se muestran una serie de imágenes con consejos para comer bien.
Excepciones	
Comentarios	Se puede ampliar las imágenes pulsando dos veces sobre la misma imagen.

022	Solicitar dietista
Actores	Paciente
Descripción	El paciente puede solicitar un dietista
Precondición	El paciente debe tener la sesión iniciada y no tiene una
Secuencia normal	1. El paciente se encuentra en la ventana principal.

	2. El paciente pulsa en <<SOLICITAR DIETISTA>> y se abre una ventana donde aparece en una *picklist* donde puede elegir al dietista que se desee. 3. El paciente puede ver la siguiente información relacionada con el dietista. a. Nombre y apellidos b. Valoración c. Descripción 4. El paciente pulsa el botón <<CONFIRMAR>>
Postcondición	Se crea una solicitud nueva que puede ver el dietista.
Excepciones	
Comentarios	

2.4 MÓDULO DIETISTA

En este módulo, se encuentran los casos de uso pertinentes al seguimiento de los pacientes por parte de los dietistas.
Este apartado es de tipo web por lo que el dietista tiene una interfaz distinta al paciente, ya que así es más cómodo para ellos a la hora de realizar su trabajo y labor.

A continuación, se encuentran los casos de uso de este módulo.

Ilustración 3:Diagrama de casos de uso Dietista

024	Asignar dieta al paciente
Actores	Dietista
Descripción	El dietista asigna una dieta a un paciente
Precondición	Debe existir el paciente, y que no tenga una dieta asignada, así como el dietista.
Secuencia normal	1. El dietista se encuentra en su ventana de inicio.

	2. Accede al apartado <<Manejar pacientes>> al pulsar el botón correspondiente. 3. Se muestra la lista de pacientes que tiene asignado el dietista. 4. Al pulsar sobre un paciente, el dietista es redirigido a otra ventana con la información de dicho paciente. 5. El dietista puede asignar una dieta a un paciente al pulsar sobre el desplegable <<Crear dieta>>.
Postcondición	Se asigna la dieta elegida al paciente.
Excepciones	
Comentarios	También se es redirigido automáticamente a la creación de la dieta cuando el dietista acepta al paciente.

025	Ver datos del paciente
Actores	Dietista
Descripción	El dietista puede ver la información de un paciente que tenga asociado, así como su progreso.
Precondición	Debe existir el paciente, y que tenga al dietista asignado.
Secuencia normal	1. El dietista se encuentra en su ventana de inicio. 2. Accede al apartado <<Manejar pacientes>> al pulsar el botón correspondiente. 3. Se muestra la lista de pacientes que tiene asignado el dietista. 4. Al pulsar sobre el botón <<Seguir progreso>>, el dietista es redirigido a otra ventana con la información de dicho paciente. 5. Aquí se muestra la información del progreso actual del paciente, así como el histórico de sus datos y progresos.
Postcondición	Se muestra la información del paciente al dietista.
Excepciones	
Comentarios	

026	Aceptar/rechazar paciente

Actores	Dietista
Descripción	El paciente ha elegido al dietista en cuestión para querer sus servicios
Precondición	Debe existir el dietista y el paciente debe haber creado la Request correspondiente.
Secuencia normal	1. El dietista se encuentra en su ventana de inicio. 2. Accede al apartado <<Aceptar pacientes>> al pulsar el botón correspondiente. 3. Se muestra la lista de pacientes que han requerido los servicios del dietista y este todavía no ha manejado. 4. El dietista puede aceptar o rechazar a cada paciente
Postcondición	-En caso de haber aceptado al paciente, se le redirige a la creación de la dieta para dicho paciente, y a partir de ese momento, ya están asignados el uno al otro. -Si se rechaza el paciente, el paciente es notificado y debe volver a elegir dietista. El dietista es redirigido a la ventana de <<Aceptar pacientes>>.
Excepciones	
Comentarios	En caso de que el dietista acepte al paciente y no le cree la dieta en dicho momento por el motivo que sea, podrá hacerlo posteriormente cuando acceda al apartado de <<Manejar pacientes>> y posteriormente al paciente en particular.

027	Eliminar paciente
Actores	Dietista
Descripción	El dietista elimina al paciente de los pacientes que él tiene asignado
Precondición	Deben estar asignados el paciente y el dietista, el uno con el otro.
Secuencia normal	1. El dietista se encuentra en su ventana de inicio. 2. Accede al apartado <<Manejar pacientes>> al pulsar el botón correspondiente.

	3. Se muestra la lista de pacientes que tiene asignado el dietista, así como un botón <<Borrar paciente>> para cada paciente que tiene asignado el dietista. 4. Cuando se pulsa sobre el botón sobre uno de los pacientes, se elimina al paciente de la lista de pacientes manejados por el dietista.
Postcondición	Se elimina el paciente de la lista de pacientes del dietista, así como la dieta si este tuviera una asignada por el dietista y el paciente podrá volver a elegir otro dietista cuando acceda a la aplicación.
Excepciones	
Comentarios	

028	Ver progreso del paciente
Actores	Dietista
Descripción	El dietista puede ver los progresos que va llevando el paciente en cuestión en lo respectivo de la dieta
Precondición	Debe existir el paciente con toda su información, y que esté asignado al dietista, y viceversa
Secuencia normal	1. El dietista se encuentra en su ventana de inicio. 2. Accede al apartado <<manejar pacientes>> al pulsar el botón correspondiente. 3. Se muestra la lista de pacientes que tiene asignado el dietista. 4. Al pulsar sobre el botón de <<Seguir progreso>> de un paciente, el dietista es redirigido a otra ventana con la información de dicho paciente. 5. En esta ventana, el dietista podrá modificar la dieta del paciente en el caso de que tenga una, o crear una nueva si no tiene, así como ver la información personal del paciente, y ver el registro de progresos que tiene el paciente.
Postcondición	Se muestra la información del progreso del paciente al dietista.
Excepciones	

Comentarios	En el caso de que el paciente no haya llevado a cabo ningún progreso todavía, se mostrará un mensaje anunciando de ello.

029	Modificar dieta del paciente
Actores	Dietista
Descripción	El dietista puede modificar la información de la dieta de un paciente que tenga asociado.
Precondición	Debe existir el paciente, y que tenga una dieta asignada, así como un dietista.
Secuencia normal	1. El dietista se encuentra en su ventana de inicio. 2. Accede al apartado <<Manejar pacientes>> al pulsar el botón correspondiente. 3. Se muestra la lista de pacientes que tiene asignado el dietista. 4. El dietista puede pulsar directamente sobre <<Modificar dieta>> para ir a la vista en la que podrá modificar la dieta del paciente, o primero en <<Seguir progreso>> y en dicha visual, tendrá nuevamente un botón de <<Modificar dieta>> 5. Al pulsar sobre <<Modificar dieta>>, se abrirá una nueva ventana en la que se muestra la información del paciente, así como un formulario con la dieta que tiene, con los campos de la dieta que el paciente ya tiene asignada rellenos en el caso de que ya lo estuvieran. 6. El dietista puede modificar los registros que él quiera de la dieta
Postcondición	Se modifica dieta del paciente en cuestión.
Excepciones	
Comentarios	Las dos primeras comidas de cada día, así como el comentario sobre cada día, son obligatorios. En el caso de que el dietista no rellene, al menos, estos dos campos, no podrá llevar a cabo la modificación de la dieta.

030	Ver perfil dietista

Actor	Dietista
Descripción	El dietista puede ver su información personal, así como dos botones para reestablecer su contraseña y darse de baja.
Precondición	El dietista debe haber iniciado sesión
Secuencia normal	1. El dietista pulsa en <<Perfil >> 2. El dietista es redirigido a una ventana en la que se mostrará la información personal que el mismo, o el administrador, rellenaron en el registro y creación de dietista, así como dos botones en los que puede realizar la acción que muestra la descripción de dichos botones.
Postcondición	
Excepciones	
Comentarios	

Capítulo 3: Tecnología empleada

En esta sección se van a especificar detalladamente las tecnologías empleadas en cada módulo del proyecto.

En primer lugar, al ser dos personas trabajando sobre un mismo proyecto se han utilizado repositorios Git que nos han permitido la labor de programar simultáneamente a través de un sistema de control de versiones.

En el caso de la aplicación Android se ha realizado la sincronización desde el mismo programa de compilación y ejecución del proyecto.

En lo referente a la aplicación Web y a su implementación, se ha usado el programa **"GitHub Desktop"** como herramienta de control de versiones. Se ha optado por usar esta aplicación ya que permite usar GitHub a través de una GUI que permite controlar y manejar las versiones de una manera más rápida y eficiente que el navegador o la línea de comandos del editor de código fuente **"Visual Studio Code (VSCode)".**

Como editor de textos se ha utilizado **"Sublime Text 3"** y **"VSCode"**. **"Sublime Text 3"** es un editor de texto muy sencillo e intuitivo y su utilidad no ha sido más que esa. En el caso de **"VSCode"** ha sido más utilizado como entorno de trabajo habitual ya que además de ser un editor de texto posee muchas integraciones y funcionalidades muy útiles tales como la visualización de la terminal.

3.1 APLICACIÓN ANDROID

Para el desarrollo de la aplicación Android se ha optado por utilidad Android Studio. Uno de los principales motivos de su elección es la inmensa cantidad de documentación que posee ya que es el entorno oficial de desarrollo para aplicaciones Android. Posee un potente editor de códigos, un sistema de compilación flexible basado en Gradle y un emulador rápido y cargado de funciones entre otros. Entre sus lenguajes de programación compatibles se encuentra Java y XML así que se ha optado por usarlos.

3.2 APLICACIÓN WEB

Para el desarrollo de la página Web y lo referente a la web, se han usado, como es normal, los lenguajes propios de esta: ***HTML5(***Handlebars****) y CSS3*** para el front end de la Web y ***JavaScript*** para el back end.

Handlebars* es un popular sistema de plantilla en JavaScript que permite crear y formatear código HTML de tal manera que se hace muy fácil la reutilización de código HTML, así como crear componentes web que se usan en varios documentos HTML con el fin de no repetir código, consiguiendo así mayor limpieza del código HTML.

Con JavaScript se ha realizado la conexión con la base de datos de Firebase, Realtime Database, pues esta base de datos ha sido desarrollada de tal manera para la Web que solo se puede acceder a ella y operar con ella a través del lenguaje de programación JavaScript.

Además, para ciertos apartados de la página Web se ha usado el Framework ***Bootstrap***. Bootstrap permite el fácil y rápido desarrollo de componentes de las páginas web, como los formularios de registro y de creación de dietas, gracias a su kit de herramientas de código abierto para desarrollos web responsive.

3.3 BASE DE DATOS

Se ha utilizado Firebase como base de datos principal. Firebase es una plataforma móvil que facilita la creación de aplicaciones móvil y web.

Firebase permite la creación de aplicaciones con un determinado dominio de forma gratuita. Una vez se ha creado la aplicación se puede acceder a diversas aplicaciones que han sido utilizadas para este proyecto.

1. Authentication: Un sistema de gestión de usuarios propio que proporciona un control total del flujo de usuarios y contraseñas en la aplicación. Además, posee un buen soporte de restablecimiento de contraseñas a través de correos electrónicos
2. Realtime Database: Es una base de datos NoSQL alojada en la nube que se ha utilizado para almacenar toda la información utilizada en la aplicación.
3. Storage: Es un gestor de archivos alojado en la nube que ha sido utilizado a la hora almacenar imágenes.

Es cierto que en ciertos casos una base de datos NoSQL puede llevar a problemas a la hora de crear relaciones entre objetos. Aun así, han sido solventados todos esos problemas a través de código ya que se ha visto más conveniente realizar ese esfuerzo a nivel de código y poder disfrutar de un servidor en tiempo real muy accesible y gratuito.

Capítulo 4: Arquitectura de la aplicación

El sistema que se ha utilizado en la aplicación ha sido la de servidor-cliente. El servidor almacena toda la información del cliente en la base de datos y el cliente accede a esos datos a través de internet.

En el lado de Android lo único que se necesita es tener la aplicación instalada en el dispositivo Android y además tener una conexión a internet. Esas dos condiciones son primordiales ya que, si no, no se tendrá la capacidad de conectarnos al servidor.

Por el lado Web, Firebase tiene un servicio de pago para el hosting de las webs. Sin embargo, al tratarse de un trabajo de fin de grado que, a pesar de tener una gran proyección de futuro a través de mejoras y continuas posibles actualizaciones, solo se puede acceder a la parte web a través de localhost. Para ello se ha usado ***Express***, que es un framework minimalista y flexible que dispone Node.js para la fácil implementación y despliegue de páginas web.

Una vez se tienen las dos partes funcionando, el servidor se encarga de la autenticación de clientes, de la persistencia de datos y el almacenamiento de imágenes. Esto es gracias a las funcionalidades nativas que tiene que Firebase que ya han sido mencionadas anteriormente en el apartado de Capítulo 3: Tecnología empleada.

Ilustración 4: Arquitectura

En la **aplicación móvil,** es donde el usuario, que quiere mejorar sus hábitos alimenticios, usará lo pertinente a ello, como es el acceso a la base de datos para recoger su información, la información

Capítulo 5: Modelo de datos

El modelo de datos utilizado en la aplicación está basado en Realtime Database de Firebase. Realtime Database utiliza una estructura basada en JSONs.

Todos los identificadores que estén relacionados con imágenes estarán almacenados en "Storage" de Firebase que se mostrará a continuación.

También cabe destacar que todos los identificadores que aparecen son generados de manera aleatoria, no siguen ningún patrón establecido.

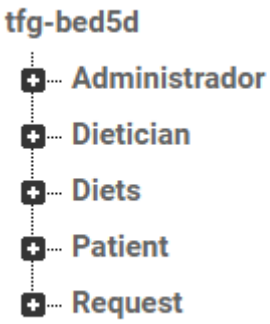


Ilustración 5:Modelo general

Principalmente se han creado 5 colecciones: ***Administrador, Dietician, Diets, Patient, Request***.

5.1 ADMINISTRADOR

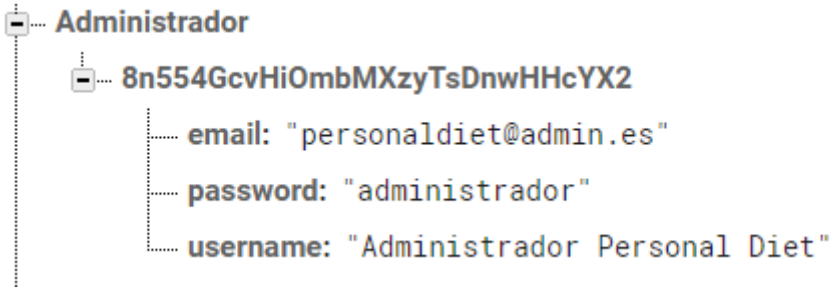


Ilustración 6: Modelo Administrador

La colección ***"Administrador"*** es la más pequeña, pero se ha decidido que debería tener una colección propia debido a que su lectura se iba a realizar en numerosas ocasiones por lo que convenia mantenerlo alejado del resto de usuarios.

En esta colección se encuentran los siguientes campos:

- *email*: email que hace referencia a la cuenta de usuario del administrador.
- *password*: contraseña que identifica al usuario.

- *username*: nombre de usuario del Administrador.

5.2 DIETISTA

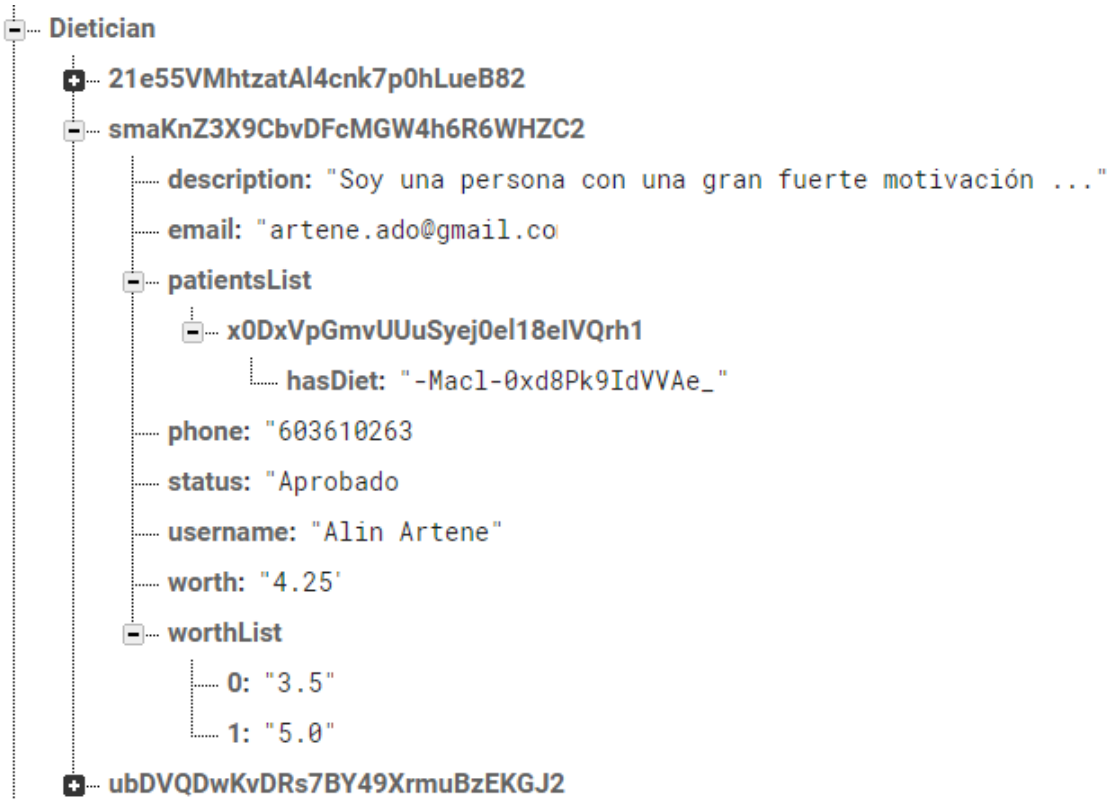


Ilustración 7 Modelo Dietista

La colección ***"Dietista"*** es una de las colecciones más extensas que existen almacenadas en la base de datos. Es el pilar de la creación de usuarios en la aplicación web, ya que los dietistas son creados por ellos mismos a través del registro que tienen en la aplicación web a su disposición o también son creados por el administrador del sistema en su interfaz para ello. Cada dietista consta del identificador único, como tienen todas las colecciones al crearse, así como una serie de datos que permiten el correcto manejo de los datos de la aplicación.

A continuación, se describen cada uno de los campos de esta colección, así como el uso y la finalidad que tienen cada uno de ellos:

- *Description:* este campo está destinado para que se rellene con una breve descripción que el dietista quiere mostrar a los pacientes que puedan elegirle, a los pacientes que ya tiene aceptados, o el administrador, al manejar dietistas. La finalidad de este campo es que, gracias a esto, el administrador pueda querer aceptarle en la aplicación, así como los pacientes ser convencidos por su descripción y así elegir sus servicios.
- *Email:* campo imprescindible para la identificación de los usuarios, el correo de usuario del administrador, al que llegarán las notificaciones de cambio de contraseña, y con el que podrá contactarse con los pacientes en caso de que quiera.

- *patientsList:* es la lista de los identificadores de los pacientes que el dietista tiene a su cargo, así como un campo en cada uno de ellos:
    - *hasDiet:* este campo sirve como identificador de la dieta que tiene dicho paciente, o "null" en caso de que no tenga ninguna dieta asignada. La finalidad de este campo es la eficiencia en el manejo de los pacientes y de las dietas de cada uno de ellos.
- *Phone:* otro campo imprescindible para la información personal del dietista, así como para la comunicación con el paciente a través de este número de teléfono, pues este será el número en el que los pacientes, a través de la aplicación móvil, puedan acceder de una manera inmediata a la conversación por WhatsApp con su dietista.
- *Status:* este campo se rellene a "Pendiente de aprobar" cuando un dietista se registra por su cuenta en la aplicación web. El dietista no podrá realizar ninguna de las funcionalidades que permite la aplicación web mas que ver su perfil, reestablecer su contraseña o darse de baja. Es el administrador del sistema el encargado de aceptar o rechazar a los dietistas. En el caso de aceptarles, ese campo cambiara su valor a "Aprobado" y es a partir de ahí cuando el dietista podrá ejercer sus servicios en la aplicación.
- *Username:* es el nombre y apellido del dietista, campo que se rellena al crear el usuario en la página web.
- *Worth:* es la media de todas las valoraciones recibidas por el dietista por sus pacientes, y la finalidad de este campo es que sirva de indicador para los pacientes que puedan querer los servicios del dietista. Tiene los valores de 0 a 5, donde 0 es una mala valoración, y 5 una excelente valoración.
- *worthList:* este campo es una lista de valoraciones que ha recibido el dietista desde que se ha registrado en la página web. Al crearse el dietista, este campo se rellena con el primer valor de la lista, con un valor de 3.5 sobre una escala de 5.

5.3 DIETAS

Ilustración 8 Modelo Dietas

Esta colección es creada exclusivamente por los dietistas desde la parte Web. Cada dieta contiene la información de cada día de la semana además de su creador y a quien lo tiene asignado.

- *Lunes-Domingo*: Cada día contiene la siguiente información

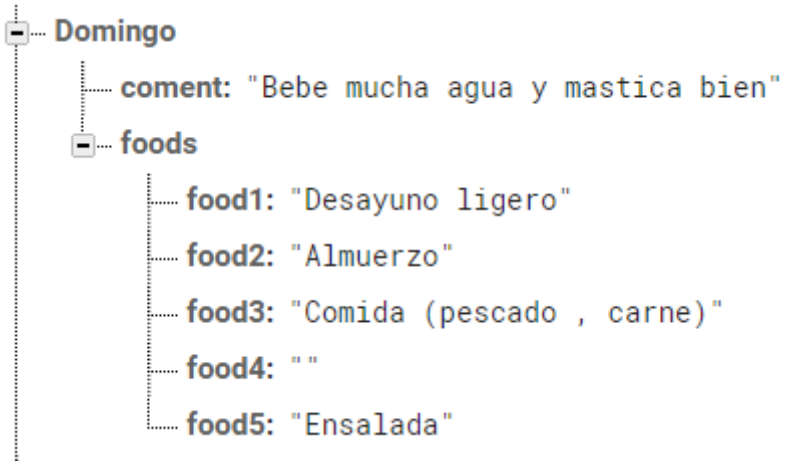


Ilustración 9 Modelo Dietas Detalle

  - *coment*: Contiene una descripción dada por el dietista exclusivamente para el paciente
  - *foods*: Contiene un total de 5 campos. Cada campo hace relación a una comida del día. Cada "foodX" contiene las indicaciones específicas de lo que debe comer o no.
- *dieticianId*: identificador que hace referencia al dietista que creó la dieta
- *patientId*: identificador del paciente al que se le ha asignado la dieta.

5.4 PACIENTE

Ilustración 10 Modelo Paciente

La colección ***"Patient"*** contiene toda la información relacionada con el paciente.

- *follow*: Hace referencia un listado de todos los seguimientos realizados por el paciente

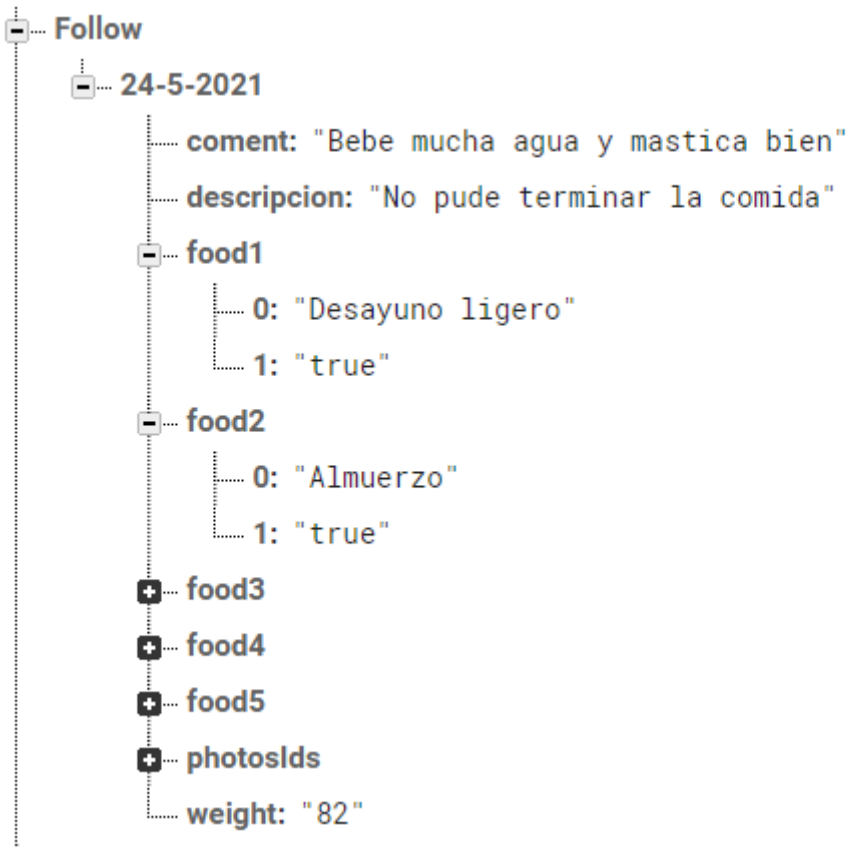


Ilustración 11 Modelo Paciente Seguimiento

  - *día*: El identificador de cada seguimiento es el día en el que se realiza.
  - coment: Comentario de la dieta que tuvo el día X.
  - descripción: Información dada por parte del paciente para que el seguimiento del paciente contenga más detalles.
  - food1-5: Lista que contiene las comidas y un booleano que indica que si siguió correctamente o no la dieta.
  - photosIds: Lista de identificadores de las fotografías almacenados en la nube que aporta el paciente.
  - weight: peso del paciente el día X.
- *activity*: Nivel de actividad deportiva del paciente
- *age*: edad del paciente
- *dietId*: Id de la dieta asignada al paciente.
- *dieticianId*: Id del dietista asignado al paciente.
- *dieticianValorated*: booleano que indica si el paciente ha valorado al dietista:
  - true: valorado
  - false: sin valorar
- *email*: correo electrónico del paciente.
- *numpics*: contador que hace referencia al número de imágenes de estado físico que ha realizado el paciente.
- *phone*: número de teléfono del paciente.
- *picIds*: lista con los identificadores de las fotos de estado físico del paciente.
- *sex*: sexo del paciente.

- username: Nombre del paciente.
- weight: peso del paciente.

5.5 PETICIONES

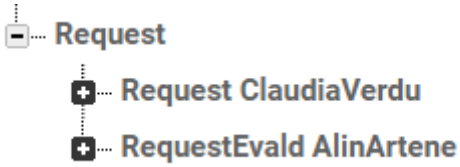


Ilustración 12:Modelo Petición

Todas las solicitudes de dietista realizadas por pacientes se almacenan en esta colección.

Los campos que guardan estas colecciones de datos son los siguientes:

- *idDietician:* es el identificador del dietista que tiene la solicitud hecha por parte del paciente de asociarse.
- *idPatient:* es el identificador del paciente que requiere de los servicios de un dietista mediante la petición.

5.6 STORAGE

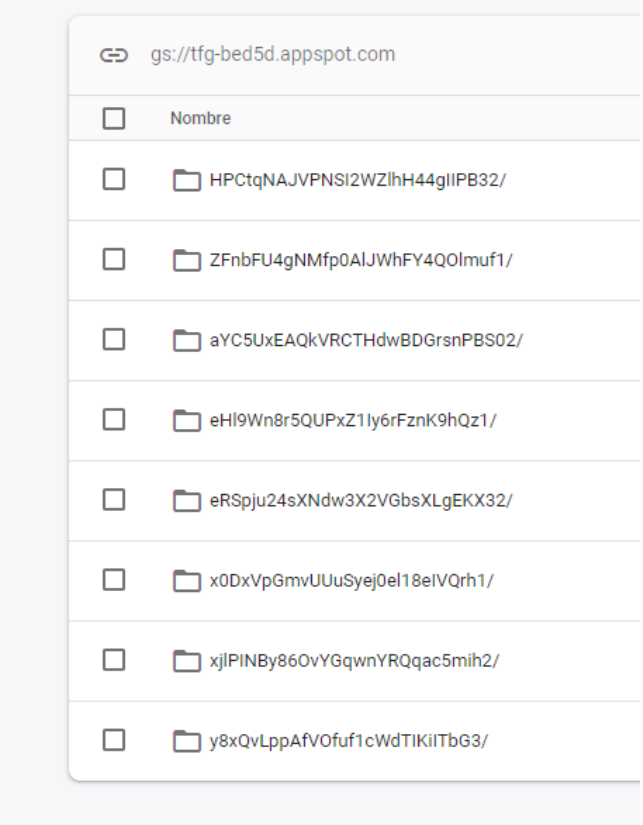


Ilustración 13 Modelo imágenes

Cada carpeta representa un paciente donde dentro tendrán toda la siguiente estructura

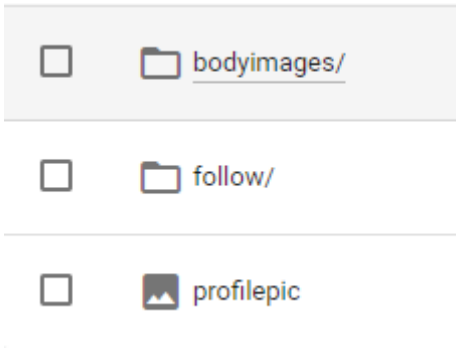


Ilustración 14 Modelo imágenes Detalles

- *bodyimages*: guarda todas las fotos del estado físico del paciente.
- *follow*: guarda todas las imágenes de todos los seguimientos realizados por el paciente. Cada seguimiento tendrá su carpeta única donde aparecerán las fotos relacionadas con ese día.

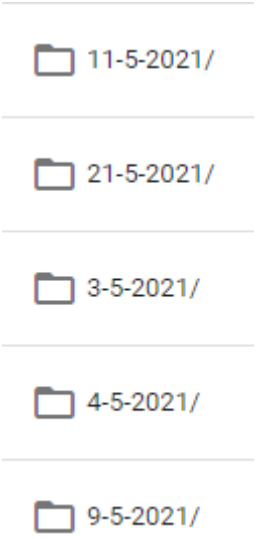


Ilustración 15 Modelo Imágenes Detalles Seguimiento

- *profilepic*: contiene la imagen de perfil del paciente.

Capítulo 6: Diseño

El diseño de una aplicación tanto web como Android tiene que ser un punto a tener en cuenta ya que es la primera imagen que un usuario recibe de la aplicación.

Para el diseño se ha considerado la facilidad de uso como el pilar principal; por lo tanto, se ha procurado hacer que la aplicación sea fácil de usar e intuitiva. Además, se ha utilizado una paleta de colores vivos y llamativos.

Como logo de la aplicación se ha optado por una fuente que resulte poco agresiva donde predominen las curvas.

Ilustración 16: Logo

Se puede ver en la imagen superior como se hace referencia a ***"Personal Diet"***. Se ha optado por este nombre después de realizar una gran lluvia de ideas para intentar encontrar un nombre que encajase bien con la idea. Al final se ha elegido ***"Personal Diet"*** ya que refleja exactamente lo que la aplicación predica, que es la posibilidad de tener dietas personalizadas hechas para el usuario.

6.1 DISEÑO DE LA APLICACIÓN MÓVIL

Android Studio ofrece un gran abanico de posibilidades ya que ofrece un diseñador ***"drag and drop"*** muy completo combinado con código ***XML*** que facilita toda la labor de diseño. Con esto en mente se ha optado por llevar un tema un juvenil y amable.

Tal y como se puede ver en las siguientes imágenes se encuentran elementos que cualquier persona podría relacionarlas con una vida saludable, y para hacerlo más juvenil se le han añadido ojos y boca.

Ilustración 17:Brócoli

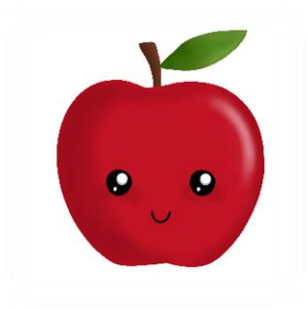


Ilustración 18:Manzana

Ilustración 19: Plátano

En la pantalla inicial se ha realizado una mezcla de los tres dibujos unificándolos con el rotulo de la aplicación tal y como se puede ver en la imagen inferior. Además, la imagen de la manzana (la imagen central) es la que se ha utilizado para que sea el icono de la aplicación.

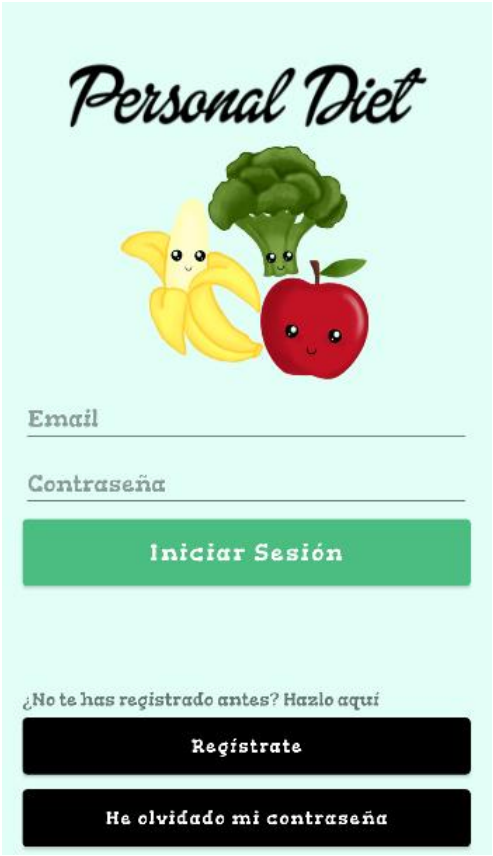


Ilustración 20 Login Diseño

La primera imagen que ve el usuario es la mostrada en la ilustración anterior. Se puede observar la composición de las frutas/vegetales mencionada antes y el tono de los colores usado.

Tal y como se puede ver en la siguiente imagen se puede observar la funcionalidad que ofrece Android Studio mencionada anteriormente.

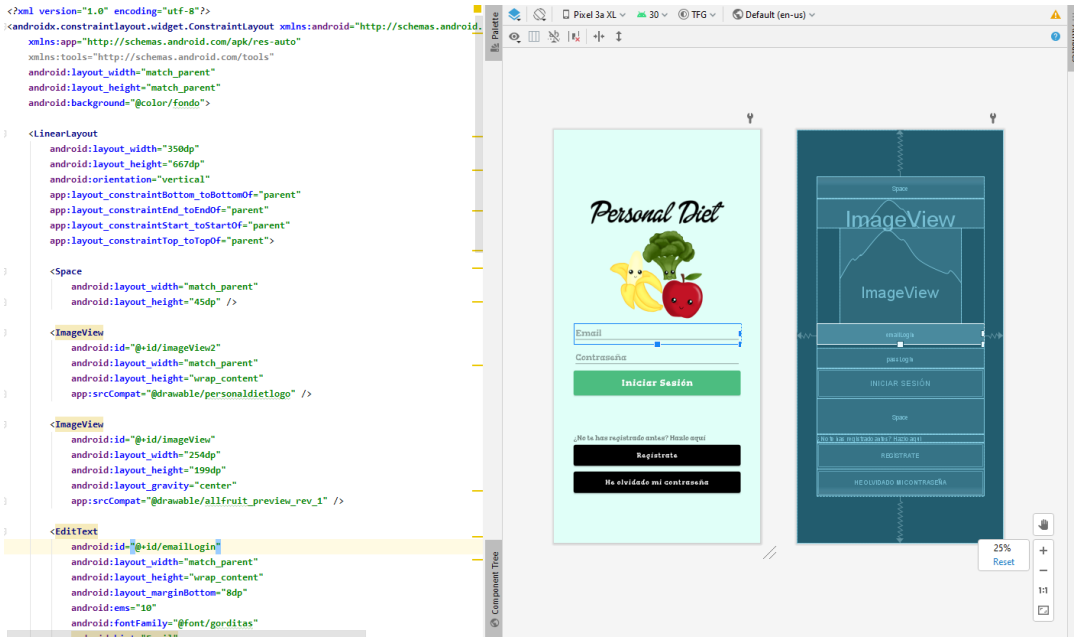


Ilustración 21:Login Android Studio

La fuente utilizada es "Gorditas" que sigue la línea llevada en el logo. A continuación, se muestran una serie de ilustraciones en las que se puede ver el estilo general que se ha llevado en la aplicación.

Ilustración 23:Ejemplo Diseño 1

Ilustración 24: Ejemplo Diseño 2

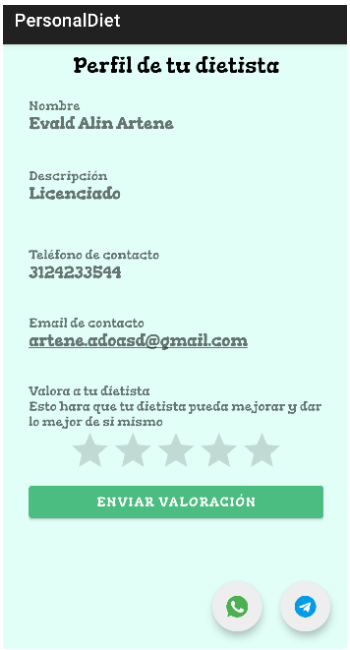


Ilustración 22: Ejemplo Diseño 3

6.2 DISEÑO DE LA APLICACIÓN WEB

En cuanto al diseño de la página web, como se ha mencionado anteriormente, se ha elegido realizar un diseño sencillo con una interfaz limpia. El objetivo de ello es que se pretende que los usuarios que usan la aplicación web la enfoquen como su herramienta de trabajo, por lo que el diseño no debe propiciar que sea costoso aprender a usar la aplicación, de ahí que sea todo muy intuitivo.

Con esta premisa en mente, los diseños para las diversas páginas web han sido realizados usando ideas que se han ido adquiriendo a lo largo de la carrera para las distintas funcionalidades que se han desarrollado, así como usado un framework de código abierto llamado ***Bootstrap*** para las distintas páginas que tienen formularios en ellas, así como para las páginas que contienen información personal de los usuarios, ya sean el administrador, el dietista, o los pacientes.

Debido a que la aplicación debe inspirar y fomentar que los usuarios de esta lleven a cabo un cambio en sus hábitos alimenticios, o que ayuden a otros a hacer eso mismo, la estructura que componen las páginas de la aplicación web es la misma para todas.

Todas las páginas tienen una barra de navegación situada en la parte superior de la pantalla, con el logo de la aplicación a un lado y la navegación que puede realizar el usuario en el otro lado, todo esto con un fondo verde claro por detrás. Debajo de esta barra de navegación, que es cambiante en función de si el usuario tiene iniciada sesión en la aplicación, y de qué tipo de usuario es, si administrador o dietista, se encuentra el contenido de la página en la que se encuentra, todo rodeado de un fondo hecho a base de imágenes de frutas, más concretamente, piñas.

Con lo detallado anteriormente, se muestra a continuación el formulario de registro para el dietista. En él se puede ver la estructura detallada, así como la información de registro, acompañado de una imagen de un desayuno saludable, algo que siempre se viene a la cabeza cuando se piensa en unos hábitos de vida saludables.

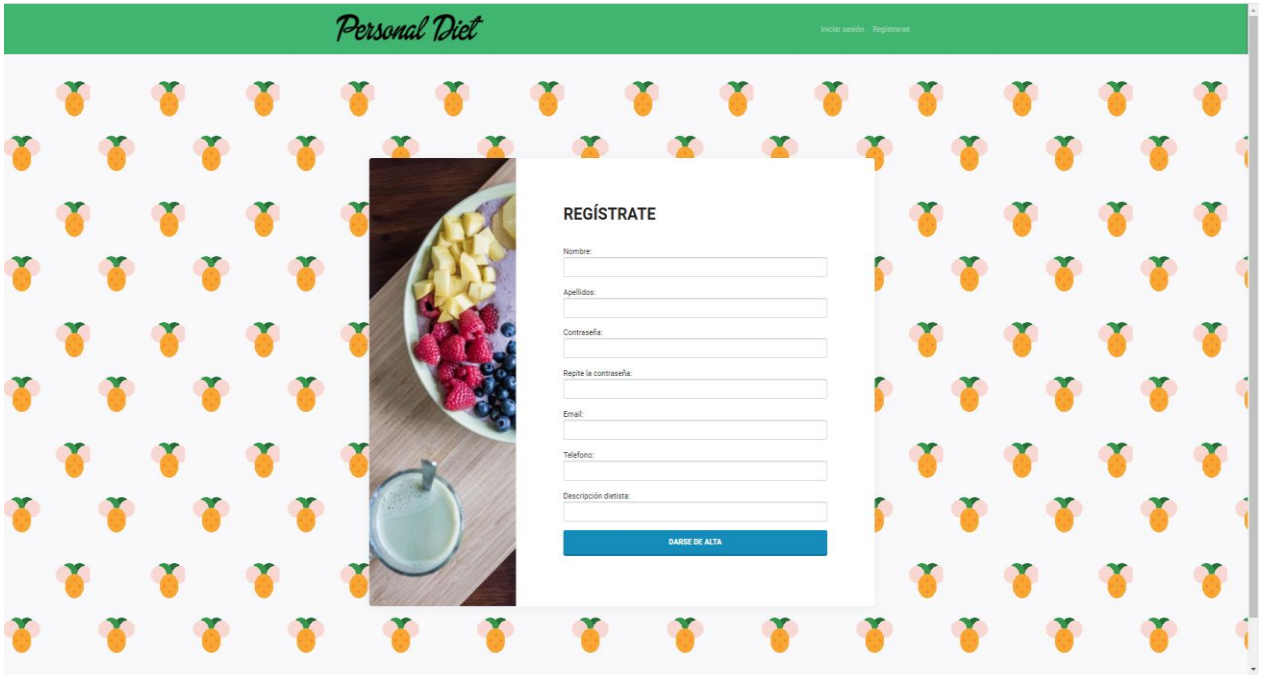


Ilustración 25 Diseño registro dietista

La siguiente imagen que se va a mostrar es la del seguimiento de uno de los pacientes que tiene el dietista asignado, en el que se muestra la información personal del paciente en cuestión, así como el progreso y la descripción que se tiene sobre el progreso diario del paciente, todo esto de manera que de un golpe de vista se pueda ver toda la información que el dietista considera relevante para el caso del seguimiento del progreso. Aquí se puede ver nuevamente la estructura que se sigue para todas las páginas, la navegación del usuario en la parte superior de la pantalla, continuado de dos secciones, una para la información del perfil del paciente, y otra en la que aparece una lista de seguimientos diarios del paciente. Aunque en este caso solo hay un seguimiento, si se tuvieran varios, saldrían uno debajo del otro, todo acompañado por un *scroll* para el caso en el que el paciente tenga muchos registros, y poder tener una navegación fácil por estos.

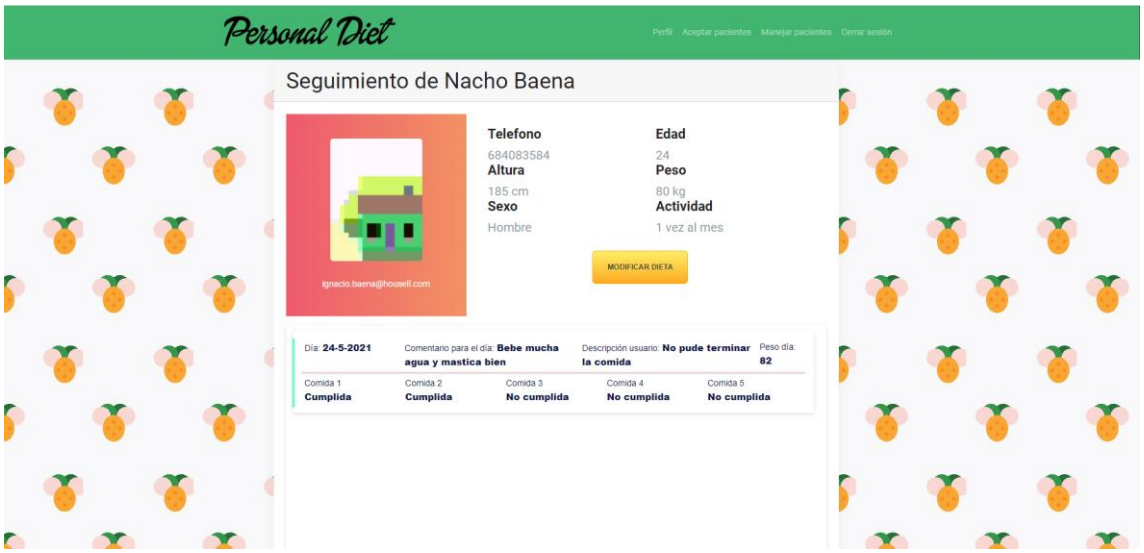


Ilustración 26 Diseño seguimiento paciente

Por último, se enseña la visual que tiene el administrador cuando accede a su apartado de manejar dietistas. En este caso se ve la navegación que el usuario administrador dispone en la parte superior de la página, y, a continuación, un listado de todos los dietistas que se encuentran en la base de datos de la aplicación, con cierta información personal que al administrador le puede interesar, así como dos botones para cada uno para realizar las acciones descritas en el propio botón.

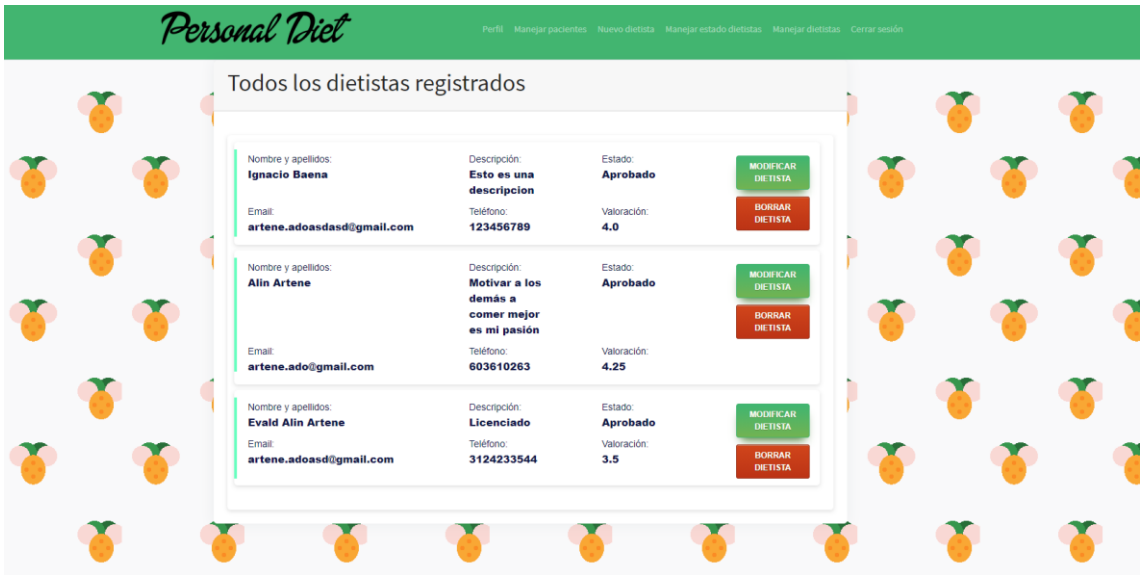


Ilustración 27 Diseño manejar dietistas Administrador

Capítulo 7: Implementación

A continuación, se explica la implementación realizada tanto en la aplicación Android como en la aplicación Web. Además, se detalla la forma en la que el cliente manda y recibe información al servidor.

7.1 IMPLEMENTACIÓN ANDROID

La implementación de la aplicación sigue el esquema que Android Studio da por defecto, compuesto por una serie directorios principales.

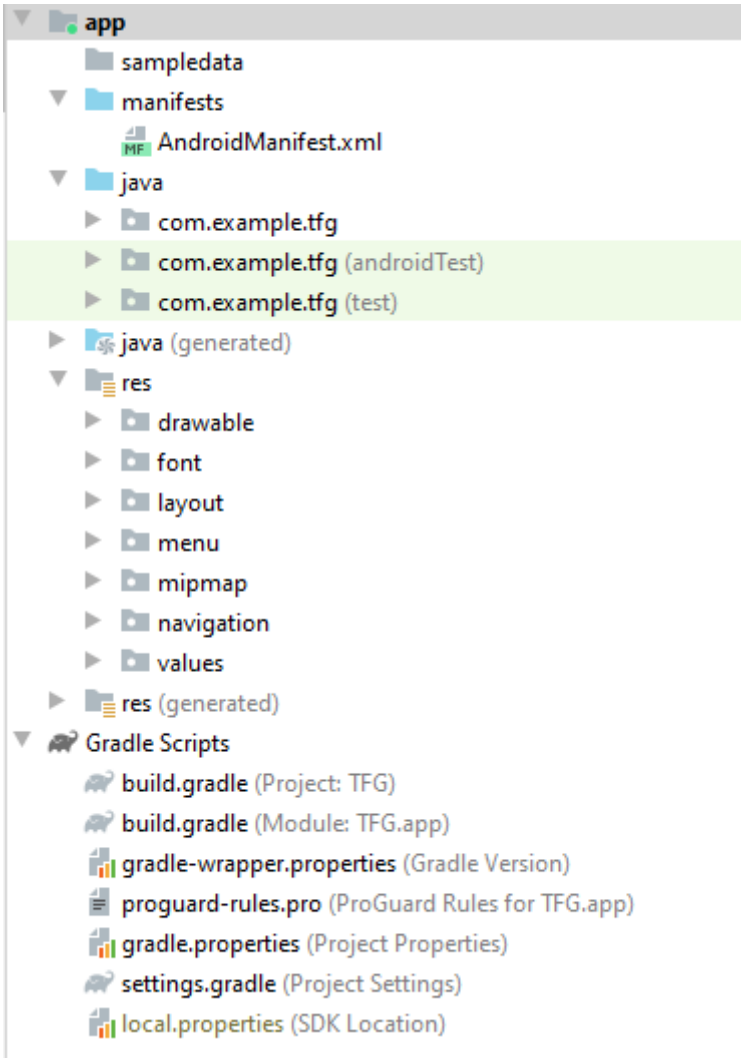


Ilustración 28 Esquema Android Studio

- manifests: En este directorio se encuentra AndroidManifest.xml. En este archivo se indica toda la información esencial que los dispositivos Android necesitan conocer para el correcto funcionamiento de las aplicaciones. En el caso del

proyecto, se solicita el acceso a internet, el acceso al reloj del móvil para poder acceder al gestor de alarmas y el acceso a la cámara.

- java: En este directorio se encuentra el groso del código fuente donde toda la lógica de la aplicación es ejecutada.
- res: Aquí se muestra toda la información relacionada a imágenes, vistas, textos, fuentes etc.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="com.android.alarm.permission.SET_ALARM"/>
<uses-feature android:name="android.hardware.camera"
```

Ilustración 29: Manifest

- Gradle Scripts: En el fichero build.gradle se declaran todas las dependencias, plugin, necesarias para que la aplicación funcione correctamente.

Las vistas de las aplicaciones son almacenadas en la carpeta res/layout. Cada layout representa una vista de la aplicación y son documentos de tipo *.xml*. Para poder visualizar esas vistas se necesita una clase java que instancie esos documentos. Cada elemento declarado en estos archivos tiene un Id el cual se puede acceder a ellos a través de la sentencia *R.Id.[id_elemento]*

```
public class MainActivity extends AppCompatActivity {
```

Ilustración 30:Main Acitivity

A continuación, se explican diversos conceptos esenciales.

- MainActivity: Es la primera clase que se ejecuta al iniciar la aplicación es un dispositivo móvil.
- AppCompatActivity : Es una clase que se implementa generalmente cuando se quiere mostrar una barra de acciones superior. Para el caso, se ha utilizado ya que se requiere de menú de acciones situado en la vista principal.

Para evitar una distorsión visual se ha optado por extender esta clase en el resto de las clases.

- OnCreate(): es una función que sobrescribe la clase AppCompatActivity la cual es la primera en ejecutarse al instanciar la clase; es decir, sirve de constructor de la vista. La sentencia *setContentView(R.layout.update_user)* construye la vista *update_user* alojada en la carpeta layout mencionada anteriormente.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.update_user);
}
```

Ilustración 31:OnCreate

- OnStart(): es la función que se lanza una vez la aplicación es visible

```
@Override
protected void onStart() {
    super.onStart();
    mAuth = FirebaseAuth.getInstance();
    mydb = FirebaseDatabase.getInstance().getReference();
    progressDialog = new ProgressDialog( context: this);
}
```

Ilustración 32: OnStart

- Intent: Cuando se desea hacer visible otra vista se crea una instancia nueva de esta clase la cual llama a la clase especificada en su constructor

```
public void goToRegister(View view) {
    Intent i = new Intent( packageContext: this, Register.class);
    startActivity(i);
}
```

Ilustración 33:Intent

- findViewById: esta función sirve de enlace entre la vista y las clases.

```
private TextView peso;
private TextView imc;
private TextView imcText;
private TextView nombre;
private ImageView imagen;

peso = findViewById(R.id.pesoView);
imcText = findViewById(R.id.imcTextView);
imc = findViewById(R.id.imcView);
nombre = findViewById(R.id.nombreView);
imagen = findViewById(R.id.imagenPerfil);
```

Ilustración 34:Views

Integración con Firebase.

Inicialmente se debe tener un proyecto creado dentro de Firebase, crear una nueva aplicación Android y seguir los sencillos pasos que nos brinda Firebase.

Una vez se hayan seguido todos los pasos correctamente la aplicación está lista.

Firebase permite integrarse en aplicaciones Android a través de una serie de sencillas sentencias. Para explicar dichas sentencias se muestran algunas de las funciones más representativas dentro de la aplicación para evitar la duplicidad de información en esta memoria.

- Realtime Database: Para instanciar a la base de datos se debe crear este atributo que hace referencia a la base de datos.

```
DatabaseReference mydb;
```

  Se hace a través de la siguiente sentencia.

```
mydb = FirebaseDatabase.getInstance().getReference();
```

  La base de datos se ramifica a través de nodos padres/hijos y para acceder a la información siempre se tiene que instanciar desde el padre.

```
mydb.child("Patient").child(id).updateChildren(map);
```

  La función child ([nombre de hijo]) hace que la referencia se posicione en ese nodo. Una vez posicionados Firebase puede añadir más funciones para hacer lo siguiente :

  - Leer: addListenerForSingleValueEvent () se posiciona en el nodo indicado y es capaz de acceder a todos atributos de ese nodo, incluso los nodos hijos de ese nodo.
  - Escribir: updateChildren (map) actualiza el nodo con la información que se contenga el mapa, si existe la clave/valor se actualiza y si no crea un atributo nuevo
  - Borrar: removeValue () elimina el nodo en el que se encuentra la referencia.

- Authentication: Para instanciar a la base de datos se debe crear este atributo.

```
FirebaseAuth mAuth;
```

  Se hace a través de la siguiente sentencia.

```
mAuth = FirebaseAuth.getInstance();
```

- Storage: Para instanciar a la base de datos se debe crear estos atributos.

```
FirebaseStorage storage;
StorageReference storageReference;
```

  Se hace a través de la siguiente sentencia.

```
storage = FirebaseStorage.getInstance();
storageReference = storage.getReference();
```

A través de storageReference se puede acceder a la base de datos de Storage la cual sigue el mismo esquema de nodos padre/hijo

Registro/Inicio de sesión de usuarios y persistencia de datos

```
private void signUp(String email, String password) {
    if (TextUtils.isEmpty(email)) {
        Toast.makeText( context: this,  text: "Se debe ingresar un email", Toast.LENGTH_LONG).show();
        return;
    }

    if (TextUtils.isEmpty(password)) {
        Toast.makeText( context: this,  text: "Falta ingresar la contraseña", Toast.LENGTH_LONG).show();
        return;
    }
    progressDialog.setMessage("Realizando registro en linea...");
    progressDialog.show();

    try {
        mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {


                    Toast.makeText( context: Register.this,  text: "Se ha registrado el usuario con el email: " + email, Toast.LENGTH_LONG).show();

                    Map<String, Object> map = new HashMap<>();
                    map.put("email", email);
                    map.put("username", "");
                    map.put("phone", "");
                    map.put("age", 0);
                    map.put("weight", 0.0);
                    map.put("height", 0.0);
                    String id = mAuth.getCurrentUser().getUid();
                    mydb.child("Patient").child(id).updateChildren(map);
                    StorageReference ref = storageReference.child(id + "/images/profilepic");
                    ref.putFile(Uri.parse("android.resource://com.example.tfg/drawable/user")) UploadTask
                            .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
                                @Override
                                public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {

                                }
                            }) StorageTask<UploadTask.TaskSnapshot>
                            .addOnFailureListener(new OnFailureListener() {
                                @Override
                                public void onFailure(@NonNull Exception e) {
                                    progressDialog.dismiss();
                                    Toast.makeText( context: Register.this,  text: "Failed " + e.getMessage(), Toast.LENGTH_SHORT).show();
                                }
                            })
                            .addOnProgressListener(new OnProgressListener<UploadTask.TaskSnapshot>() {
                                @Override
                                public void onProgress(UploadTask.TaskSnapshot taskSnapshot) {

                                }
                            });
                    finish();
                } else {

                    Toast.makeText( context: Register.this,  text: "No se pudo registrar el usuario ", Toast.LENGTH_LONG).show();
                }
                progressDialog.dismiss();
            }


        });
    } catch (Exception e) {
        Toast.makeText( context: Register.this, e.getMessage(), Toast.LENGTH_LONG).show();
        progressDialog.dismiss();
        // [END sign_in_with_email]
    }

}
```

Ilustración 35:SignUp Android

A la hora realizar el registro de un usuario en la base de datos se precisa de la función createUserWithEmailAndPassword (email, password). Esta función crea un usuario a través de la API de Authentication de Firebase.

En particular, en el código desarrollado, si la creación del usuario se realiza con éxito se procede a crear un registro adicional en Realtime Database con ciertos datos proporcionados por el cliente, tales como el email, y datos no ofrecidos por el cliente como el id que es generado de manera aleatoria. Ese Id generado simula la labor de una clave primaria en la base de datos.

```
private void signIn(String email, String password) {
    if (TextUtils.isEmpty(email)) {
        Toast.makeText( context: this,  text: "Se debe ingresar un email", Toast.LENGTH_LONG).show();
        return;
    }

    if (TextUtils.isEmpty(password)) {
        Toast.makeText( context: this,  text: "Falta ingresar la contraseña", Toast.LENGTH_LONG).show();
        return;
    }
    progressDialog.setMessage("Realizando registro en linea...");
    progressDialog.show();
    // [START sign_in_with_email]
    mAuth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        Toast.makeText( context: Login.this,  text: "Login correcto: " + email, Toast.LENGTH_LONG).show();
                        String id = mAuth.getCurrentUser().getUid();
                        mydb.child("Patient").child(id).addValueEventListener(new ValueEventListener() {
                            @Override
                            public void onDataChange(@NonNull DataSnapshot snapshot) {
                                if (snapshot.exists()) {
                                    String nombre = snapshot.child("username").getValue().toString();
                                    if (nombre.equals("")) {
                                        Intent intent = new Intent( packageContext: Login.this, FirstLogin.class);
                                        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                        startActivity(intent);
                                        finish();
                                    } else {
                                        Intent intent = new Intent( packageContext: Login.this, ProfileMenu.class);
                                        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                        startActivity(intent);
                                        finish();
                                    }
                                }
                            }
                            @Override
                            public void onCancelled(@NonNull DatabaseError error) {}
                        });
```

Ilustración 36:SignIn Android

Una vez registrado el usuario correctamente en la base de datos, el cliente puede acceder a la aplicación. Para ello se utiliza la función signInWithEmailAndPassword (email, password).

Si la función es completada con éxito a través el usuario accede a la pantalla principal de la aplicación si ya había entrado con anterioridad. En caso de que fuese la primera vez se le llevará a vista diferente.

Si la función no se completa con éxito se muestra un mensaje Toast en la vista.

Subir imágenes.

Para almacenar archivos de image en Storage se accede a la galería del dispositivo móvil con la siguiente sentencia.

```
public void SelectImage(View view) {

    Intent intent = new Intent();

    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    startActivityForResult(
            Intent.createChooser(
                    intent,
                    title: "Select Image from here..."),
            PICK_IMAGE_REQUEST);
}
```

Ilustración 37:Seleccionar imagen Codigo

Una vez elegida la imagen se captura la imagen la función sobrescrita onActivityResult (…) que recibe la información de la función startActivityForResult que se muestra anteriormente.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == PICK_IMAGE_REQUEST && resultCode == RESULT_OK
            && data != null && data.getData() != null) {
        filePath = data.getData();
        try {

            System.out.println(" filePath");
            System.out.println(filePath);
            Bitmap bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), filePath);
            imagen.setImageBitmap(bitmap);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Ilustración 38: OnResult Activity

Si la imagen es correcta en el atributo filePath se instancia la dirección del archivo dentro del dispositivo móvil. A su vez para mostrar la imagen dentro de la aplicación se genera un mapa de bit y se instancia la ImageView a través de ella con image.setImageBitmap().

Una vez se tenga seleccionada la imagen, se sube la imagen a Storage con el siguiente código.

```
public void uploadImage(View view) {
    AlertDialog.Builder dialog = new AlertDialog.Builder( context: UploadPhoto.this);
    dialog.setTitle("¿Es correcta la foto?");
    dialog.setPositiveButton( text: "Si", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            if (filePath != null) {
                final ProgressDialog progressDialog = new ProgressDialog( context: UploadPhoto.this);
                progressDialog.setTitle("Uploading...");
                progressDialog.show();
                String id = mAuth.getCurrentUser().getUid();
                String key = UUID.randomUUID().toString();
                StorageReference ref = storageReference.child(id + "/images/bodyimages/" + key);
                ref.putFile(filePath) UploadTask
                        .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
                            @Override
                            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                                progressDialog.dismiss();
                                Toast.makeText( context: UploadPhoto.this,  text: "Uploaded", Toast.LENGTH_SHORT).show();
                                if (numpic == 0) {
                                    picIds.set(0, key);
                                } else
                                    picIds.add(key);
                                Integer suma = numpic + 1;
                                Map<String, Object> map = new HashMap<>();
                                map.put("numpics", suma);
                                map.put("picIds", picIds);
                                mydb.child("Patient").child(id).updateChildren(map);
                            }

                        }) StorageTask<UploadTask.TaskSnapshot>
                        .addOnFailureListener(new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                progressDialog.dismiss();
                                Toast.makeText( context: UploadPhoto.this,  text: "Failed " +  e.getMessage(), Toast.LENGTH_SHORT).show();
                            }
                        })
                        .addOnProgressListener(new OnProgressListener<UploadTask.TaskSnapshot>() {
                            @Override
                            public void onProgress(UploadTask.TaskSnapshot taskSnapshot) {
                                double progress = (100.0 * taskSnapshot.getBytesTransferred() / taskSnapshot
                                        .getTotalByteCount());
                                progressDialog.setMessage("Uploaded " + (int) progress + "%");
                            }
                        });


            }
        }
    });
    dialog.setNegativeButton( text: "No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { dialog.dismiss(); }
    });
    AlertDialog alertDialog = dialog.create();
    alertDialog.show();


}
```

Ilustración 39: Subir imagen Firebase Android

Realtime Database y Storage no tienen una comunicación directa por lo que ha sido necesario crear una relación a través de listas de Ids para una persistencia real de imágenes entre pacientes.

Tal y como se ha mencionado antes, en Storage cada usuario tiene un directorio propio. Cuando el usuario sube una foto se genera un numero aleatorio que es el identificador tanto en Realtime Database como Storage. Como la relación es 1 – N , es decir , 1 único paciente puede tener varias imágenes, pero no se tiene una base de datos relacional, se

mapea la información con una lista de Ids generados aleatoriamente dentro del nodo de pacientes.

Gestión del menú

En la pantalla principal aparece un menú desplegable a la parte superior derecha. Ese menú se encentra en una parte distinta al resto de layout y posee una nomenclatura especifica.

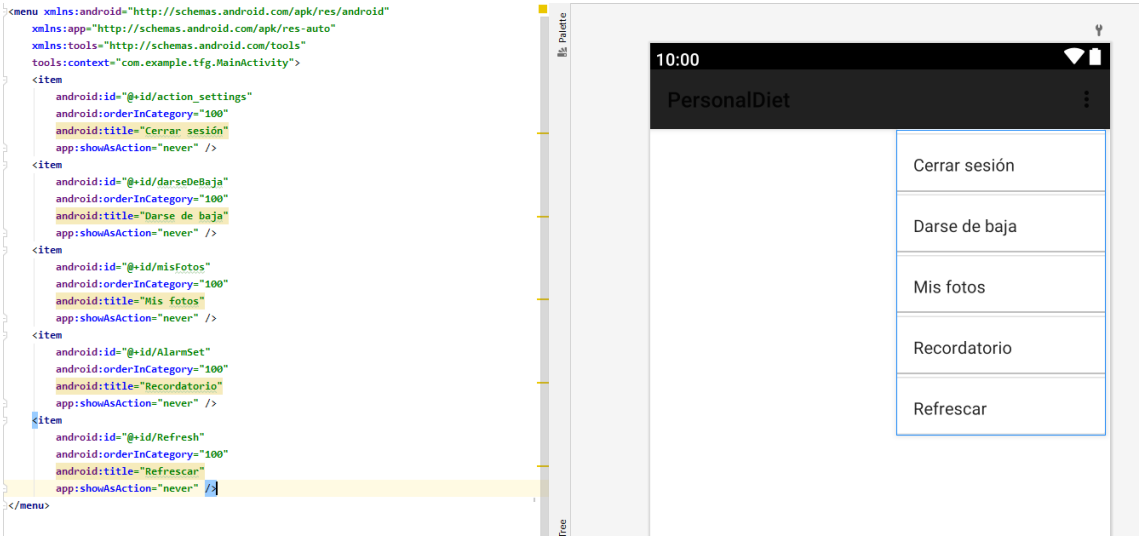


Ilustración 40: Diseño Menú

Una vez diseñado el menu, existen funciones especificas para su manejo dentro de las clases java. En la vista en la que se desea colocar el menu se utiliza lo siguiente:

- OnCreateOptionMenu() : hace la misma funcion de OnCreate anteriormente mencionado pero especificamente para el menu
- MenuInflater : sirve para hacer visible el menu dentro de la vista en la que se está.
- onOptionsItemSelected: hace la funcion de enrutador de las opciones disponibles dentro del menu

```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
    switch (item.getItemId()) {
        case R.id.action_settings:
            cerrarSesion();
            return true;
        case R.id.darseDeBaja:
            darseDeBaja();
            return true;
        case R.id.misFotos:
            misFotos();
            return true;
        case R.id.AlarmSet:
            setAlarm();
            return true;
        case R.id.Refresh:
            refresh();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Ilustración 41: Código Menú

7.2 IMPLEMENTACIÓN WEB

En este apartado, se presenta como se ha optado por llevar a cabo la implementación web. Para ello, se presenta a continuación el esquema de directorios y archivos que forman la aplicación web, así como una descripción de cada uno de ellos.

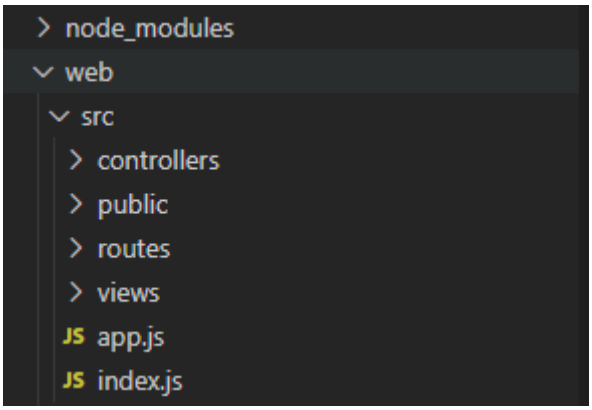


Ilustración 42 Directorios aplicación web

- Directorio *node_modules:* como ya se ha mencionado anteriormente, la aplicación web se ha implementado usando Node.js, por lo que, para el correcto funcionamiento de esta, se han usado una serie de módulos que proporciona este entorno en tiempo de ejecución. Estos módulos se gestionan a través de ***npm***, que es el sistema de gestión de paquetes por defecto para Node.js. Los módulos son archivos JavaScript, desde muy sencillos a muy complejos, y organizados en uno o más archivos, que implementan por si mismos diversas funcionalidades y que se pueden usar a través de Node.js.
  Estos módulos de Node.js, se deben instalar en la propia aplicación web, cada uno mediante su comando oportuno, y luego se tienen que "importar" en el proyecto mediante la orden require('nombre_módulo'). A continuación, se ilustra esta orden, a la vez que la ejecución del módulo express, que se almacena en una constante *app* para su posterior uso.

```
const express = require('express'); //importar el modulo express
const morgan= require('morgan'); //middleware morgan
const hbs = require('express-handlebars'); //modulo express handlebars
const path= require('path');//permite trabajar con los directorios
const app = express(); //ejecutar el modulo
```

Ilustración 43 Web: Módulos Node.js

Entre los que se han usado en el proyecto, los más destacados son:

- *Firebase:* este módulo es imprescindible para el correcto uso y funcionamiento de la base de datos que se ha usado, ya que provee de las herramientas e infraestructura necesaria para la implementación de la conexión con dicha base de datos. En el caso de Firebase, se deben proporcionar una serie de atributos cuando se quiere inicializar la aplicación, atributos como el nombre del proyecto en la base de datos Firebase, así como el dominio de dicho proyecto en Firebase, entre otros.
-

```
//FIREBASE
// Firebase App (the core Firebase SDK) is always required and
// must be listed before other Firebase SDKs

var firebase = require("firebase/app");

// Add the Firebase products that you want to use
require("firebase/auth");
require("firebase/firestore");

var firebaseConfig = {
    apiKey: "AIzaSyDwQ_xg8NmGEmPwj3Oj0sRrE_R_8qhC2cc",
    authDomain: "tfg-bed5d.firebaseapp.com",
    databaseURL: "https://tfg-bed5d.firebaseio.com",
    projectId: "tfg-bed5d",
    storageBucket: "gs://tfg-bed5d.appspot.com/",
    messagingSenderId: "1081286364712",
    appId: "1:1081286364712:web:3782a5e9c03bc46bde6f1d",
    measurementId: "G-TNC92QC113"
  };

  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
  const auth = firebase.auth();
```

Ilustración 44 Web: Integración Firebase

- *Express:* es un sencillo framework para el correcto desarrollo de aplicaciones web, pues proporciona una serie de herramientas para el fácil despliegue de estas. Es el marco de servidor estándar de facto para Node.js.
    - *Express-Handlebars:* es un motor de plantillas hecho para *express* muy potente y simple que sirve para poder renderizar páginas web en el lado del cliente con información del servidor. También permite usar condicionales simples e iteradores en el código HTML, así como registrar los llamados *"helpers"*, que permiten implementar estructuras como condicionales más complejos en el HTML.
- *Morgan:* es un *middleware*, un software cuya lógica permite el intercambio de información entre aplicaciones, paquetes de programas, redes, hardware o sistemas operativos. Este middleware sirve para simplificar el tratamiento de las solicitudes HTTP hechas al servidor usado.
- *Path:* es un módulo que permite tratar y transformar las rutas de los archivos que se usan/importan en la aplicación web, y así hacer de su uso mucho más sencillo.

- Dentro del directorio */web/src*, se tienen los siguientes directorios/archivos

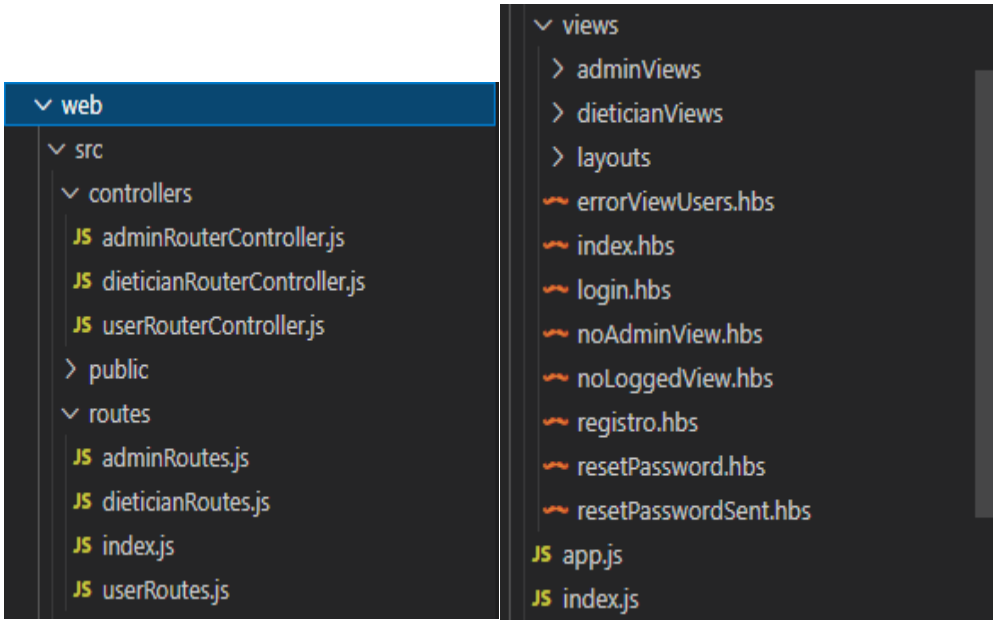


Ilustración 45 Web: Directorio web

  - *Controllers:* en este directorio se encuentran los métodos que se usan en la aplicación web, y está dividido en función de quien hace uso de estos. Los tres archivos que componen este directorio se detallan a continuación.
    - *adminRouterController.js*: este archivo contiene todos los métodos pertinentes a la implementación exclusiva de los casos de uso del administrador en la aplicación web.
    - *dieticianRouterController.js:* al igual que el archivo que tiene los métodos que implementan los casos de uso del administrador, este archivo hace lo propio con los del dietista.
    - *userRouterController.js:* este archivo contiene los métodos referentes a los casos de uso que ambos dos actores que intervienen en la aplicación web, el administrador y el dietista, hacen uso. Estos casos de uso son las funcionalidades pertinentes al acceso a la aplicación web, al inicio de sesión de los usuarios que hacen uso de esta, el cierre de esta, así como el registro y la renderización de las distintas vistas comunes a los actores, entre otros.
  - *Public:* esta carpeta contiene los directorios y archivos de código, así como imágenes y demás recursos utilizados por la aplicación web para dar estilo a las distintas páginas web. Entre los archivos de este directorio, aquí se encuentran las hojas de estilo usadas para las distintas vistas que ofrece la página web, así como los recursos(imágenes), así como los archivos y directorios usados por *Bootstrap* para la implementación de ciertas vistas.

- *Routes:* en este directorio se encuentran los manejadores de las rutas que se han usado para la aplicación web a través de los módulos que se usan gracias a Node.js. Las rutas se encargan de redirigir las peticiones HTTP que se hacen, de manera que estas se puedan separar e implementar modularmente con el fin de realizar su manejo de una manera más sencilla.
  A continuación, se muestra uno de los archivos que se encarga de manejar las rutas que son propias de las acciones realizadas por el dietista.

Ilustración 46 Web: Manejador de rutas Dietista

En la ilustración de arriba, se puede ver como se manejan las distintas peticiones HTTP, tanto GET como POST, que se hacen al servidor. Cuando se quiere acceder a la ruta /perfilDietician, el manejador del dietista realiza esta petición ejecutando la función perfilDietician () que se encuentra en el controlador dieticianRouterController.

- *Views:* En esta carpeta, se encuentran todas las visuales de las distintas páginas web que componen la aplicación web. Estas vistas han sido divididas en varios directorios, así como otras que se han dejado en el directorio principal
  - *adminViews:* son las distintas vistas que se muestran en la página web una vez que se ha iniciado sesión como administrador, y solo este puede verlas.

- *dieticianViews:* son las distintas vistas que se muestran en la página web una vez que se ha iniciado sesión como dietista, y solo este puede verlas.
- *Layouts:* en este directorio se encuentran los fragmentos de código HTML que se rehúsan para no repetir código en cada una de las vistas de cada uno de los actores de la aplicación web. En esta carpeta se encuentran las barras de navegación posibles que hay, la barra de navegación para un usuario que ha entrado en la aplicación web, pero todavía no ha iniciado sesión, la barra de navegación propia del usuario administrador, y la barra de navegación del dietista; así como un archivo llamado *main.hbs*, cuyo código implementan todas las demás páginas web y que se compone de las distintas directivas que el navegador necesita para procesar el archivo como un archivo HTML.

- *Index.js:* este archivo es el encargado de lanzar toda la aplicación web y procesar toda la información, pues es el archivo que, mediante línea de comandos de Visual Studio se hace lanzar la aplicación web al puerto configurado para ello. El comando que se usa para el despliegue de la aplicación es ***npm run dev.***
-

Ilustración 47 Web: index.js

- *App.js:* este archivo contiene todas las instrucciones y directivas que hacen la posible integración de todos los componentes que se usan en la aplicación web. En este archivo se encuentra la inicialización de los distintos módulos de Node.js que se usan, así como las configuraciones pertinentes para cada uno de ellos, como es la integración de Firebase en el proyecto una vez instalado el módulo acorde. Aquí se establece en que puerto se lanza la aplicación web para poder usarla, así como la declaración de los distintos manejadores de rutas. Toda esta información se exporta en un nuevo módulo, llamado módulo app, que es usado por el archivo *index.js* mencionado arriba.
- *Package.json:* es un archivo que se crea automáticamente al crear un proyecto usando Node.js que contiene toda la información que se conoce sobre el proyecto centralizada en un simple fichero de texto en formato JSON.

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "dev": "nodemon web/src/index.js"
},
```

Ilustración 48 Web: package.JSON

En este archivo se ha establecido que, al usar el comando mencionado anteriormente (***npm run dev)*** para el despliegue de la aplicación, se ejecute lo que está en el campo ***"dev",*** lo que hace que se ejecute el archivo index.js con *nodemon*, que es una herramienta que sirve para monitorizar los cambios en el código de la aplicación haciendo, automáticamente, que el servidor se lance una y otra vez tras cada cambio.

A continuación, se muestra una traza del programa, la traza correspondiente a uno de los casos de uso del dietista, que es cuando accede este a su vista de *"Manejar pacientes"*, con su debida explicación.

En primer lugar, el dietista pulsaría, en la barra de navegación propia de los dietistas, en el botón "Manejar pacientes". Esto lo puede realizar desde cualquiera de sus vistas, pues se recuerda que la barra de navegación se encuentra en todas las vistas que se muestran una vez que se ha iniciado sesión en la aplicación como dietista.

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark border-bottom">
    <div class="container">

      <a class="navbar-brand" href="/indexDietician"><img src="/personaldietlogo.png"></a>
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav"
        aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav ml-auto">
          <li class="nav-item" >
            <a class="nav-link" href="/perfilDietician" >Perfil</a>
          </li>
          <li class="nav-item" >
            <a class="nav-link" href="/aceptarPacienteView" >Aceptar pacientes</a>
          </li>
          <li class="nav-item" >
            <a class="nav-link" href="/manejarPacientes" >Manejar pacientes</a>
          </li>
          <li class="nav-item" >
            <a class="nav-link" href="/logout" >Cerrar sesión</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
```

Ilustración 49 Web: Implementación Barra Dietista

Cuando el dietista hace clic sobre dicho botón, esto lanza una petición GET al sistema a la ruta *"/manejarPacientes"*.

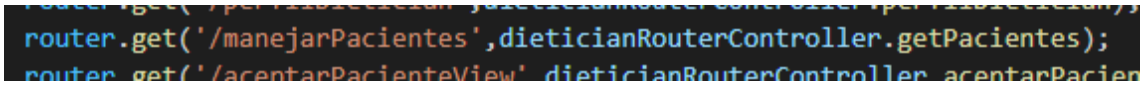


Ilustración 50 Web: petición /manejarPacientes

Esta petición, mediante el manejador de rutas del dietista, hace que se ejecute la función *"getPacientes ()"*.

```
//DEVUELVE TODOS LOS PACIENTES QUE TIENE EL DIETISTA A SU CARGO EN LA BD
function getPacientes(request, response) {
    var user = firebase.auth().currentUser;
    if (user) {
        getEstadoDietista(user.uid);
        var dieticianId = user.uid;
        if (estadoDietista) {
            db.ref('Patient').orderByChild('dieticianId').equalTo(dieticianId).on('value', function (snapshot) {
                const data = snapshot.val();
                return response.render('./dieticianViews/manejarPacientes', { patient: data });
            })
        }
        else {
            return response.render('./dieticianViews/noStatusDietician');
        }
    }
    else {
        return response.redirect('../noLoggedView');
    }
}
```

Ilustración 51 Web: función getPacientes ()

Esta función, lo primero que hace es comprobar que hay un usuario con la sesión iniciada en la aplicación. Si no tuviera iniciada sesión, se redirige al usuario a una vista en la que se le notifica de ello y se le insta a iniciar sesión. En caso de que si tuviera sesión iniciada, se comprueba que el estado del dietista en la base de datos sea "Aceptado", es decir, que el administrador del sistema le ha aceptado en la aplicación web. Si estuviera "Aceptado", se realiza la siguiente instrucción:

```
db.ref('Patient').orderByChild('dieticianId').equalTo(dieticianId).on('value', function (snapshot) {
    const data = snapshot.val();
    return response.render('./dieticianViews/manejarPacientes', { patient: data });
})
```

Ilustración 52 Web: Lectura de datos Firebase

Esta es la lectura de datos en Firebase de los pacientes asignados al dietista, con lo que se renderiza la vista web con toda la información en la constante *data.*

```
{{> navbarLoggedDietician }}

<div class="col-md-6 offset-md-3">
    <div class="card">
        <div class="card-header">
            <h1>Pacientes administrados por ti</h1>
        </div>
        <div class="card-body">
            {{#if patient}}

            <div class="slds-scrollable_y-ManejarPaciente scroll-ManejarPaciente">
                {{#each patient}}
                <!-- <lightning-card class="carta"> -->

                <div class="card-Paciente-ManejarPaciente">
                    <div class="columnaFilas-ManejarPaciente">


                        <div class="filaDatos-ManejarPaciente">
                            <div class="columnaDatos-ManejarPaciente">
                                <p class="notas-text-primera-ManejarPaciente">Nombre y apellidos:</br> <span
                                        class="notas-text-abajo-ManejarPaciente">{{username}}</span></p>

                            </div>
                            <div class=" columnainfoCorta-ManejarPaciente">
                                <p class="notas-text-primera-ManejarPaciente ">Teléfono: </br><span
                                        class="notas-text-abajo-ManejarPaciente">{{phone}}</span>
                                </p>
                            </div>
                            <div class=" columnainfoCorta-ManejarPaciente">
                                <p class="notas-text-primera-ManejarPaciente ">Altura: </br><span
                                        class="notas-text-abajo-ManejarPaciente">{{height}}</span></p>
                            </div>
```

Ilustración 53 Web: Código vista manejar pacientes dietista

En esta visual, la primera directiva sirve para instanciar la barra de navegación propia de los dietistas. Posteriormente, se tiene un condicional que nos permite ver si la lectura de datos hecha en Firebase nos ha dado algún resultado, y si así fuera, se itera sobre los datos enseñando la distinta información de cada uno de los pacientes que se han leído.

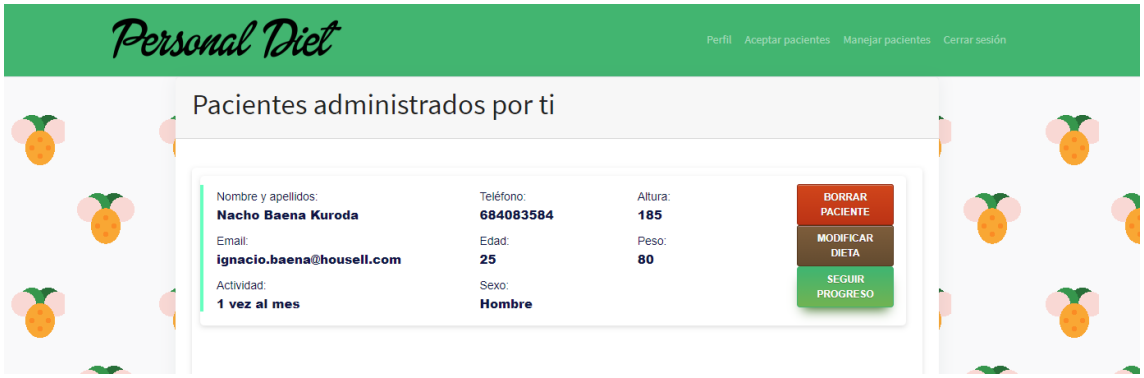


Ilustración 54 Web: Vista manejar pacientes dietista

Este sería el resultado final de la vista de la página web de *"Manejar pacientes"*.

Capítulo 8: Conclusiones y trabajo futuro

8.1 CONCLUSIONES

La aplicación, en esencia, está orientada al intercambio de información entre dos usuarios, uno que usa la aplicación desde un dispositivo Android, el paciente, y otro que usa la aplicación desde la aplicación web, el dietista. Esta información está estructurada en una dieta, así como la información personal pertinente de cada uno de los dos extremos para favorecer la correcta comunicación entre ambos, así como el correcto desarrollo del cumplimiento de la dieta.

En cuanto a las ventajas, se tiene una aplicación que una gran mayoría de usuarios puede usar, pues hoy en día casi cualquier persona dispone de un dispositivo Android en su bolsillo, y así poder mejorar sus hábitos. Por otra parte, el acceso a la aplicación web, si estuviera alojada en un servidor web, también sería muy accesible para los dietistas que quieran usar la aplicación con una motivación vocacional o profesional. Ambas dos aplicaciones han sido hechas de manera que sean muy intuitivas para el usuario, no dando lugar a error en el desempeño en las mismas.

Respecto a las limitaciones, principalmente, son dos. La primera es que, por el momento, los usuarios que quisieran usar la aplicación móvil pero no tuvieran un móvil con Android, no pueden usarla, así como el requerimiento de un servidor web donde alojar la aplicación web para ser usada por los usuarios de esta en este formato. Por otra parte, como casi cualquier aplicación, se requiera de una cierta masa crítica, tanto de usuarios pacientes como de usuarios dietistas, para poder explotar las funcionalidades de la aplicación. Esto es ya que, en caso de escasez de usuarios pacientes, los dietistas que existen en la aplicación requerirían de más usuarios para poder enfocar la aplicación de manera laboral/profesional. A su vez, en caso de haber muchos pacientes y pocos dietistas interesados en la aplicación, los dietistas que haya no darían a basto, por lo que estos pacientes dejarían de estar interesados en la aplicación móvil y dejarían de usarla.

8.2 TRABAJO FUTURO

A continuación, se van a detallar las posibles mejoras o funcionalidades nuevas que se podrían añadir en versiones futuras.

Para ambas aplicaciones.

- Integración con Telegram: Poder conectar a los clientes y al dietista a través de grupos de Telegram administrados por el dietista.

- Tablón de anuncios: Hacer que el dietista pueda poner indicaciones generales o consejos a todos los pacientes que siga, y que estos puedan visualizarlo en su ventana principal.
- Notificaciones: Poder recibir notificaciones en la aplicación cuando haya cambios o noticias nuevas.

Para Android

- Modo horizontal: Hacer un soporte total para la vista horizontal.
- Modo noche completo: Es cierto que la aplicación soporta el modo noche, pero no de la forma y los colores deseados.
- Soporte para más versiones Android: Hacer que la aplicación funcione en versiones más antiguas y así poder llegar a un público más extenso.
- Logros: Crear algún método de gamificación para premiar al paciente si está haciendo un buen trabajo.
- Mas indicadores: Además del IMC, añadir más resultados estadísticos.

Para Web

- Visualizar todas las imágenes de los pacientes: Poder visualizar todas las imágenes subidas por los pacientes que el dietista tenga asignado para tener un mayor *feedback.*
- Alojamiento en servidor: A través de un servicio de hosting tener la aplicación web disponible para todo el mundo.
- Plantillas de dietas: Tener un gestor de plantillas de dietas para que el dietista en cuestión pueda usarlas de base a la hora de crear sus dietas.
- Dietas más personalizadas: Tener un editor de dietas más completo con mayor número de opciones.

Chapter 8: Conclusions and future work

8.1 CONCLUSIONS

The app, in essence, is geared towards exchanging information between two users, one who uses the app from an Android device, the patient, and another who uses the app from the web app, the dietitian. This information is structured in a diet, as well as the relevant personal information of each of the two extremes to favor the correct communication between both, as well as the correct development of compliance with the diet.

As for the advantages, you have an application that a large majority of users can use, because nowadays almost anyone has an Android device in their pocket, and thus be able to improve their habits. On the other hand, access to the web application, if hosted on a web server, would also be very accessible to dietitians who want to use the application with a vocational or professional motivation. Both applications have been made in a way that is very intuitive for the user, not resulting in error in the performance in them.

About the limitations, there are mainly two. The first is that users who would like to use the mobile application but do not have a mobile with Android, can not use it, as well as the requirement of a web server where to host the web application to be used by users of this in this format. On the other hand, like almost any application, a certain critical mass is required, both patient users and dietitians, to be able to exploit the functionalities of the application. This is because, in case of shortage of patient users, the dietitians that exist in the application would require more users to be able to approach the application in a work/professional way. In turn, if there are many patients and few dietitians interested in the application, the dietitians that there are would not give enough, so these patients would stop being interested in the mobile application and would stop using it.

8.2 FUTURE WORK

Below, we will detail any improvements or new features that could be added in future releases.

For both applications.

1. Integration with Telegram: To be able to connect customers and the dietitian through Telegram groups managed by the dietitian.
2. Bulletin board: Make it possible for the dietitian to give general directions or advice to all patients who follow, and for them to be able to view it in their main window.
3. Notifications: Be able to receive notifications in the application when there are new changes or news.

For Android

1. Landscape mode: Make full support for the landscape view.
2. Full Night Mode: It is true that the application supports Night Mode, but not of the desired shape and colors.
3. Support for more Android versions: Make the application work in older versions and thus be able to reach a wider audience.
4. Achievements: Create some gamification method to reward the patient if they are doing a good job.
5. More indexers: In addition to the BMI, add more statistical results.

For Web

1. Visualize all the images of the patients: To be able to visualize all the images uploaded by the patients that the dietitian has assigned to have a greater *feedback.*
2. Server hosting: Through a hosting service have the web application available to everyone.
3. Diet templates: Have a diet template manager so that the dietitian in question can use them as a base when creating their diets.
4. More personalized diets: Have a more complete diet editor with more options.

Capítulo 9: Aportaciones individuales

En este apartado trata de explicar las aportaciones individuales de cada uno de los integrantes del proyecto. Para el desarrollo del proyecto, ha sido necesaria la colaboración continua de ambos dos por la estrecha relación necesaria de las dos aplicaciones que han realizado cada uno.

9.1 EVALD ALIN ARTENE

En el desarrollo referente a la memoria, en primer lugar, cuando se desarrollaron los casos de uso y alcance de la aplicación, se dividió el trabajo de manera equitativa. Cuando se tuvo el feedback del tutor del proyecto, fue el encargado de llevar a cabo la revisión de todos los apuntes que nos hizo respecto a la memoria y establecer y desarrollar la estructura de esta. En segunda instancia, en el desarrollo final de la memoria, se ha encargado de realizar todo lo relativo a la documentación de la aplicación web.

En lo relativo al desarrollo e implementación de la aplicación, el trabajado ha sido dividido en dos, por lo que, a este respecto, se ha encargado de todo el desarrollo de la aplicación web y lo pertinente a esta.

La primera dificultad que se encontró fue el encontrar una base de datos que fuera fácil accesible y comunicable con la aplicación móvil, por lo que, para ello, se ha tenido la ayuda y la recomendación del tutor, recomendando Firebase.

Firebase, para funcionar y poder integrarlo en una aplicación web, requiere de Node.js. Aunque es cierto que a lo largo de la carrera se enseñan conocimientos sobre desarrollo web, no había usado Node.js y no había explorado las herramientas de este para la creación y despliegue de servidores web. La mayor dificultad se encontró al principio de la implementación del proyecto, cuando se tuvo que crear el proyecto web y realizar el despliegue de la aplicación web mediante Node.js, así como la integración de la base de datos. Para realizar esto, se ha usado en la numerosa documentación que existe sobre Firebase, así como videos en Internet donde se explican los distintos conceptos que han hecho falta para entender y usar Node.js y la integración de Firebase en el proyecto. Lo siguiente fue aprender a usar los manejadores para las peticiones que se hacen al servidor, así como leer y guardar la información con Firebase de manera que no haya inconsistencias en dicha información. Por último, también se realizaron los diseños, así como implementaciones de las distintas páginas web, teniendo varias iteraciones para ello y descartando ciertos diseños en el proceso de mejora y revisión continua.

9.2 IGNACIO BAENA KURODA

A continuación, se procede a explicar cuál ha sido la aportación de Ignacio Baena en este proyecto.

En primer lugar, fue el encargado de la creación de los repositorios donde se iba a mantener el proyecto. En segundo lugar, se ha encargado de entender cómo era el funcionamiento de Firebase ya que era la primera vez que manejaba ese entorno de trabajo. También creó el proyecto dentro de Firebase para que la parte del servidor funcionase correctamente. Tras crearlo fue necesario dividir parte del trabajo. Firebase posee módulos específicos para Android y para Web y al ser dos personas se decidió dividir el trabajo en dos.

En este caso se encargó del módulo Android y para el desarrollo fue necesario aprender Android de cero ya que no se tenía ningún conocimiento previo respecto al desarrollo de aplicaciones móvil.

Aun habiendo trabajado en partes distintas, se ha tenido que colaborar muy conjuntamente y de manera muy precisa. Esto es debido a que a la hora de leer y escribir datos en la base de datos se debía que tener en cuenta cómo se leían y cómo se escribían en la base de datos ya que en el mayor de los casos es información compartida.

En lo que respecta la memoria, se ha encargado de representar y explicar todo lo relacionado con la parte Android. El resto de los capítulos han sido repartidos de forma equitativa.

Bibliografía

1. Gorditas - Google Fonts
2. Qué es Gradle y sus características | OpenWebinars
3. The European Food Information Council : Food facts for healthy choices (eufic.org)
4. Telegram - Wikipedia, la enciclopedia libre
5. karunstha/android-profile-ui: A template for the profile page of an app. (github.com)
6. Agrega Firebase al proyecto de Android (google.com)
7. Firebase Android - Actualizar Datos en Realtime Database #3 - YouTube
8. Firebase Android - Login y cierre de sesión de usuario con Firebase Authentication - YouTube
9. Using multicursor in Android Studio (kevinpelgrims.com)
10. 10 consejos para alimentación saludable durante la cuarentena o aislamiento (COVID-19) (eufic.org)
11. Image Slider in Android using ViewPager - GeeksforGeeks
12. Fotomontajes efectos, tarjetas de cumpleaños y marcos para fotos. - Fotoefectos
13. Stack Overflow en español
14. How to Retrieve Image from Firebase in Realtime in Android? - GeeksforGeeks
15. Diagram Maker | Software de Diagramas en Línea | Creately
16. Editor de Fotos | BeFunky: Editar Fotos Gratis Online
17. Iconos vectoriales gratis - SVG, PSD, PNG, EPS y fuente de iconos - Miles de iconos gratis. (flaticon.es)
18. Como instalar Firebase para Node.js
19. Que es Node.js y como descargarlo
20. Que es Handlebars y cómo usarlo
21. Introducción a Express/Node
22. Firebase-RealtimeDatabase
23. Firebase-RealtimeDatabase-Admin
24. Firebase-RealtimeDatabase-Storage
25. Firebase-Autentication con JavaScript
26. Firebase-Firestore CRUD con JavaScript y HTML
27. Firebase Functions REST API CRUD
28. API REST con Firebase Web
29. Curso intensivo de apps de Node.js en Firebase

Apéndice

Manual de instalación

En este último capítulo se encuentra un manual de instalación que tiene como objetivo guiar a los usuarios en el proceso de instalación y manual de usuario que explica todas las funcionalidades de las aplicaciones.

Aplicación Android

Para poder utilizar la aplicación será necesario instalar el archivo APK que se proporciona en el repositorio compartido de este proyecto o bien en el dispositivo móvil Android o bien en un emulador de Android.

Esta aplicación está lista para correr en la versión Android 11 y como mínimo es necesario tener la versión 7 de Android para hacerla funcionar.

Para emular la aplicación se recomienda BlueStacks. Todos los enlaces necesarios para la instalación se encuentran en al final de este apartado.

Ilustración 55:BlueStacks instalador

Una vez descargado el instalador, se procede a ejecutar. Cuando se complete la instalación se procede a configurar e instalar la APK en el emulador correctamente para tener una buena experiencia de usuario a tres de los siguientes pasos.

1. Rotar la pantalla. En la barra lateral derecha se encuentra este icono el cual gira la pantalla y la posiciona en una posición vertical.

Ilustración 56:Rotar (BlueStacks)

2. Cambiar resolución: a través el menú de ajustes señalada con este icono  .Una vez hechos los cambios tal y como muestran la ilustración, se validan y se reinicia el emulador.

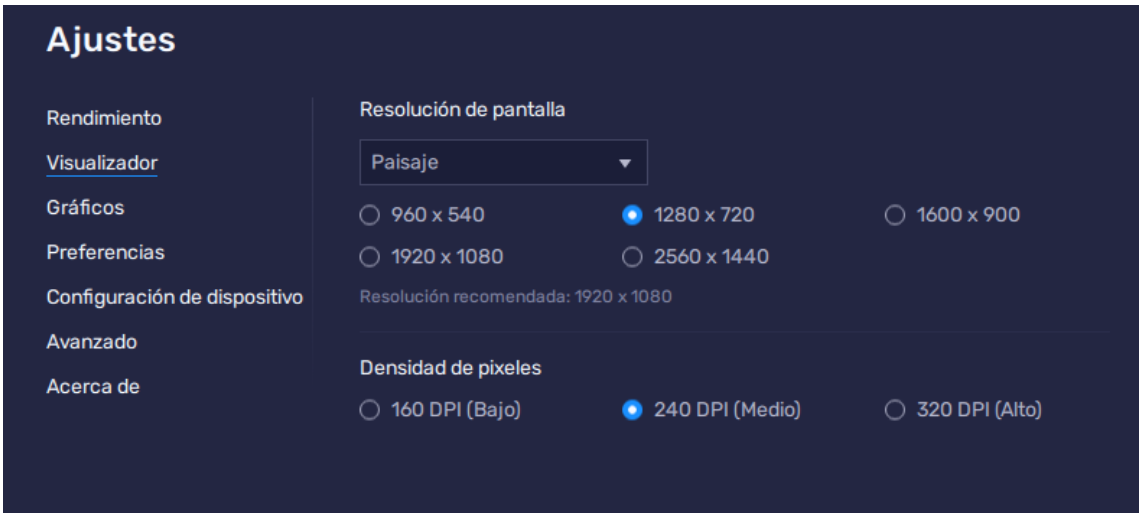


Ilustración 57:Ajustes (BlueStacks)

3. Instalar la APK:
    a. Accede a la aplicación Administrador de multimedia y al explorador de archivos del ordenador pulsando en el botón “Importar desde Windows”. Una vez hecho eso selecciona el archivo APK y valida todos los siguientes pasos.

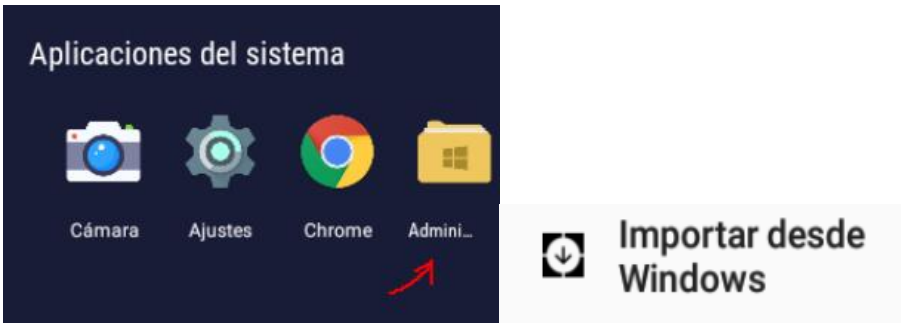


Ilustración 58:Importar desde Windows

b. Pulsa en el icono  que aparece en el apartado “Archivos importados” y acepta todos los siguientes pasos

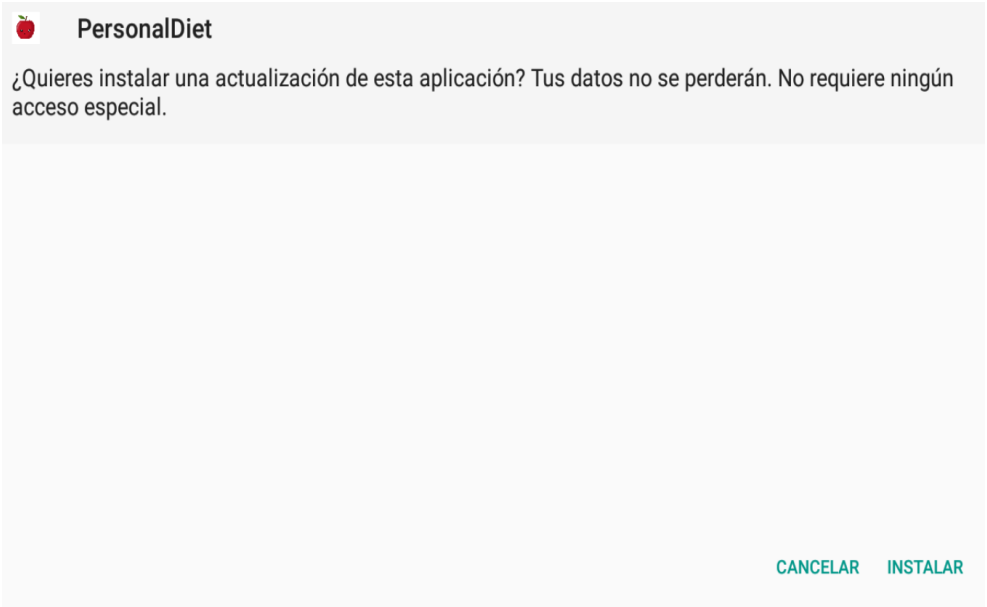


Ilustración 59:Instalar PersonalDiet

Emulador: [BlueStacks - El emulador de Android más rápido para PC y Mac |100% seguro y GRATIS]

APK: Se encuentra dentro del directorio del trabajo de fin de grado

Aplicación Web

Para la aplicación web, el proceso de instalación, por el momento, es un pelín más complejo que el requerido para el uso de la aplicación en el móvil. Esto es debido a que, al tratarse de una aplicación web, se requiere que dicha aplicación esté alojada en algún servidor en Internet. Como ya se ha mencionado anteriormente, a efectos del TFG, esto no se ha considerado oportuno, por lo que para para poder usar la web, se debe hacer lo siguiente:

- Tener instalado Node.js: esto se hace descargando la versión correspondiente de Node.js en el ordenador desde la página oficial.
- Una vez instalado Node.js, se necesitan instalar una serie de paquetes, por lo que se ha de ejecutar, desde la consola de comandos de Windows, el siguiente comando para cada uno de los paquetes que se usan en la aplicación:

  npm install -g <nombre_módulo>

  Los paquetes que instalar son los siguientes:

  - express
  - express-handlebars
  - morgan
  - firebase-admin
  - nodemon
  - body-parser
- Una vez instalados dichos módulos, se debe dirigir al directorio que contiene el proyecto, y en dicho directorio, usar el comando: ***npm run dev.***
- Tras usar el comando del punto anterior, debería salir lo siguiente

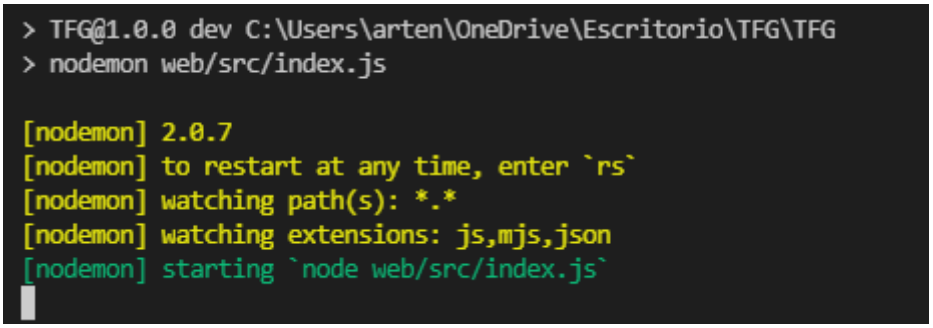


Ilustración 60 Instalación Web

- En este momento, ya se podrá usar el navegador para poder acceder a la aplicación web, por lo que se escribirá en la barra de búsqueda *localhost:4000*, y ya se tendría la aplicación desplegada, apareciendo así la página principal de la aplicación web.

Manual de usuario

En este apartado se detallará y explicará, paso por paso, los procedimientos que un usuario que nunca ha usado la aplicación puede realizar en ***PersonalDiet,*** tanto en la aplicación Android como en la aplicación web.

Aplicación Android

Registro, inicio de sesión y solicitud de dieta.

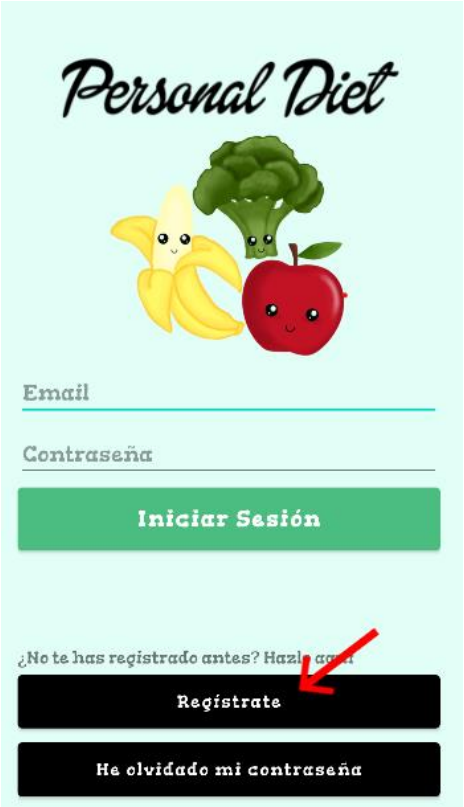


Ilustración 61: Login Android

Ilustración 62:Registro Android

Rellena los campos requeridos para crear la cuenta (email y contraseña). Si todo es correcto se redirige a la pestaña principal.

En caso de olvidar la contraseña puede recuperarla pulsando en “He olvidado mi contraseña”

Ilustración 63: Restablece tu contraseña Android

Si el email se encuentra en la base de datos de PersonalDiet recibe un correo con un link que redirige a un formulario para restablecer la contraseña

Hola:

Haz clic en este enlace para cambiar la contraseña de PersonalDiet de tu cuenta ignacio.baena@housell.com.

https://tfg-bed5d.firebaseapp.com/__/auth/action?mode=resetPassword&oobCode=O7fzqgLUShciuD8bmul_4jwkbpf7WIV1IOQ7spGp9XIAAAF53EGmRQ&apiKey=AIzaSyCRrsY0Luu-5xitIu1ZN9W2IoAVKkSxl1Y&lang=es

Si no has solicitado este cambio, ignora este correo electrónico.

Gracias,

El equipo de PersonalDiet

Ilustración 64 Email restablecer contraseña

Primer inicio de sesión.

La primera vez que un paciente ingresa en la aplicación es obligatorio rellenar los siguientes campos mostrados en la imagen y cuando los valide será redirigido a la vista principal.

Al ser un paciente nuevo el único botón disponible es “SOLICITAR DIETISTA“ ya que no tiene ni dietista ni dieta asignada.

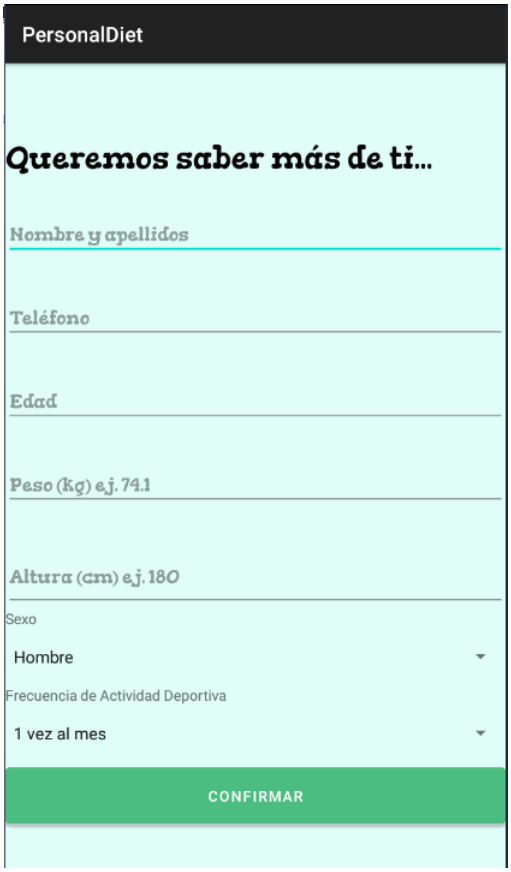


Ilustración 65: Página principal Android Android

Ilustración 66:Primer inicio de sesión

A la hora de solicitar se pueden visualizar todos los dietistas que hay en ese momento más toda la información relacionada a ellos.

Una vez elegido el dietista que se desee se crea una petición la cual el dietista debe aceptar o denegar. En cuanto el dietista acepte y cree una dieta personalizada los botones “MI DIETA” y “SEGUIMIENTO DE DIETA” se habilitan.

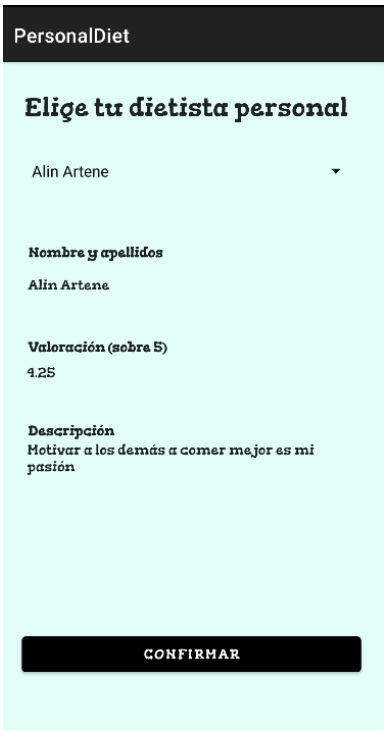


Ilustración 67: Solicitar dietista Android

Vista general de la pantalla principal

Una vez se tiene la dieta y el dietista asignado, se enumeran las funcionalidades. Los números representan las funcionalidades

1. Cerrar sesión: El paciente cierra su sesión dentro de la aplicación y vuelve a la pantalla de login.
2. Darse de baja: El paciente puede darse de baja si lo desea tras una doble confirmación.
3. Mis fotos: El paciente accede a su galería de fotos de estado físico.
4. Recordatorio: El paciente puede establecer una alarma de forma rápida y sencilla en la que pesarse
5. Refrescar: Vuelve a refrescar la vista general
6. Actualizar peso: El paciente puede actualizar su peso
7. Mi dieta: el paciente puede visualizar su dieta. Se profundiza más en este apartado más adelante.
8. Seguimiento de dieta: el paciente realiza un seguimiento de la dieta.

9. Consejos para comer bien: se despliega una serie de imágenes con consejos saludables que sirven de ayuda.
10. Ajustes de usuario: el paciente puede actualizar sus datos, así como cambiar su foto de perfil.
11. Ayuda: se muestra una imagen de ayuda que recuerda al paciente las posibilidades que tiene

Darse de baja

Si te das de baja, tu email y todas las imágenes que tengas subidas serán eliminadas de la base de datos para siempre.

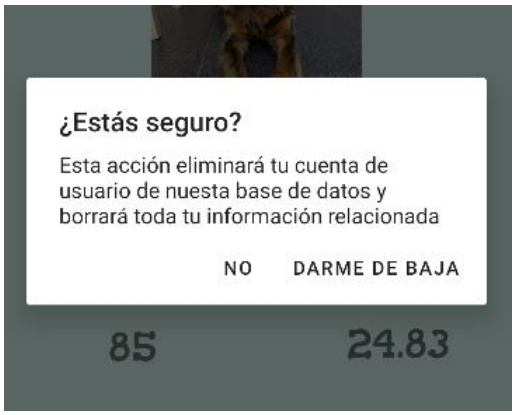


Ilustración 69: Darse de baja

Mis fotos

En este apartado el paciente puede visualizar todas las fotos de su estado físico que desee. Pulsando en “SUBIR FOTO” se abre otra vista donde podrá elegir la foto que desee de su galería.

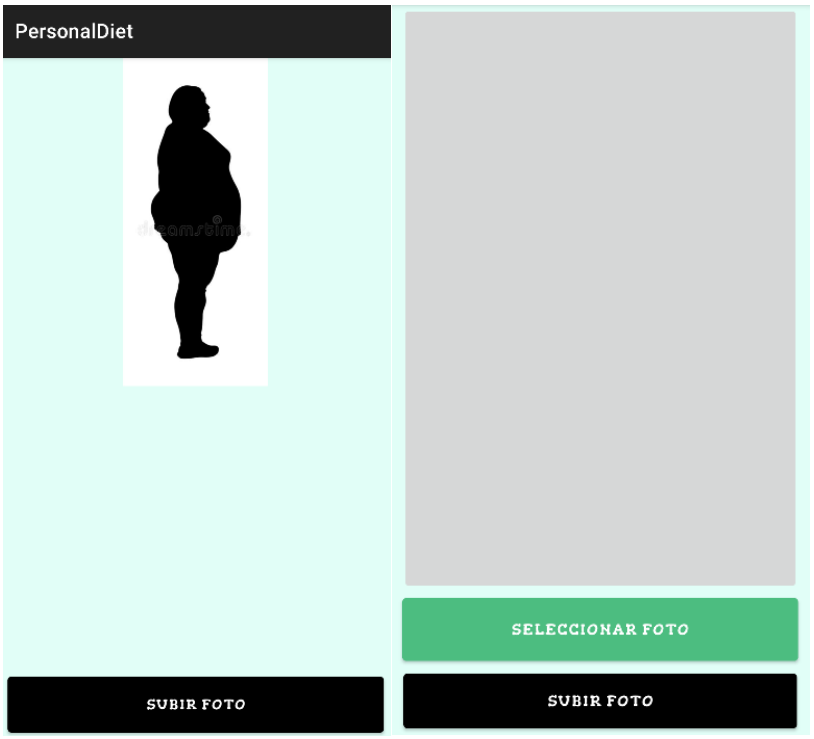


Ilustración 70: Mis fotos

Ilustración 71: Seleccionar foto 1

Para elegir la foto pulsa en “Seleccionar Foto” y una vez elegida pulsa en “Subir foto”. Pulsa en “Si” en la pregunta y la imagen habrá sido subida con éxito.

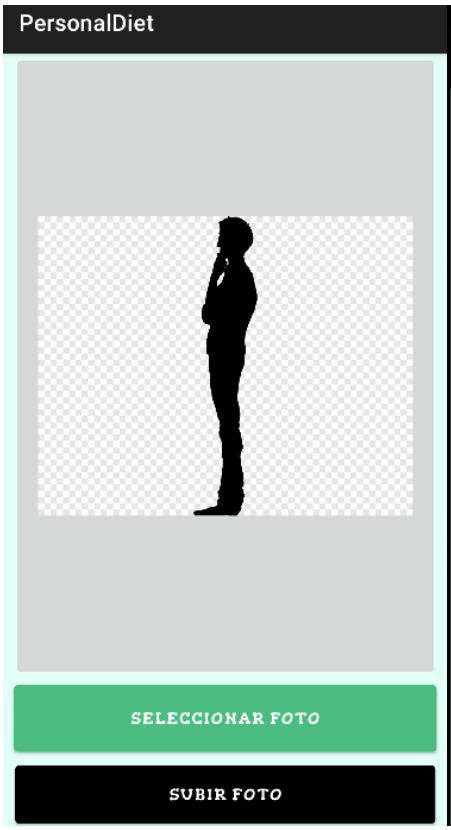


Ilustración 72: Seleccionar foto 2

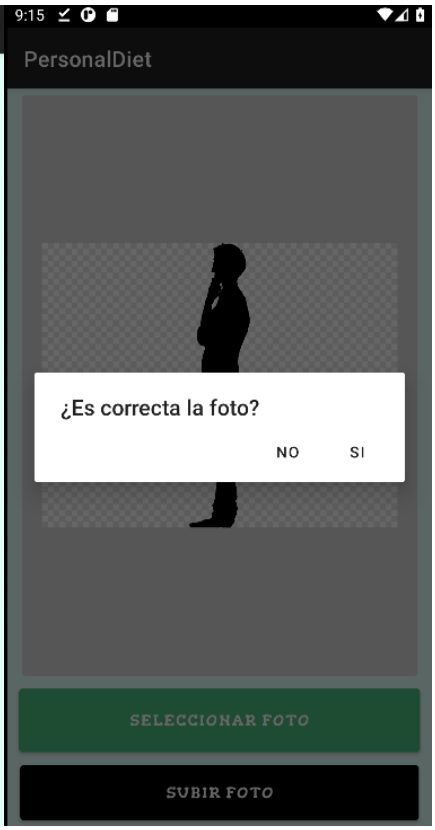


Ilustración 73: Seleccionar foto 3

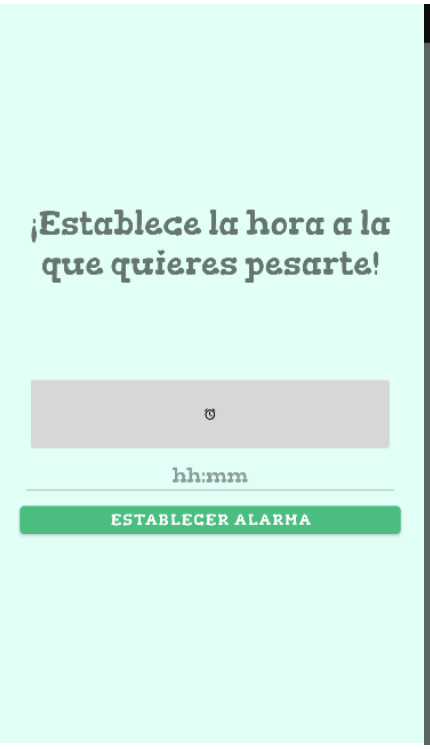
Finalmente, todas las imágenes que se suben pueden ser eliminados realizando una pulsación larga encima de la imagen.

Ilustración 74: Eliminar foto Mis Fotos

Recordatorio

Pulsando en el botón grande con el icono de un reloj se despliega un reloj donde se puede establecer una hora a la hora que pesarte.

Establecida la hora pulsa en "ESTABLECER ALARMA" y automáticamente la aplicación se cierra y se abre la interfaz del reloj del sistema donde se puede ver la alarma creada.



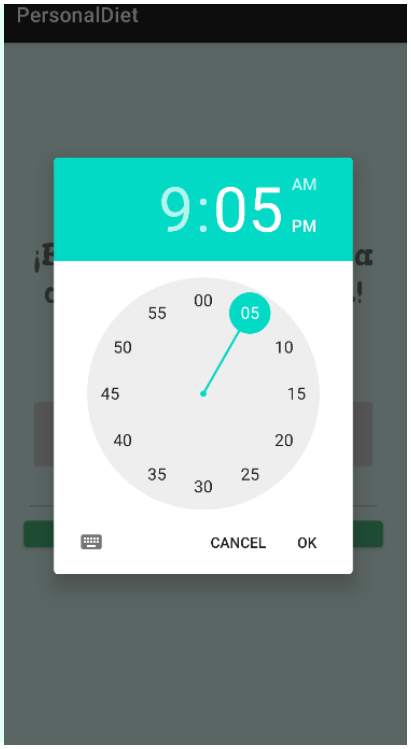
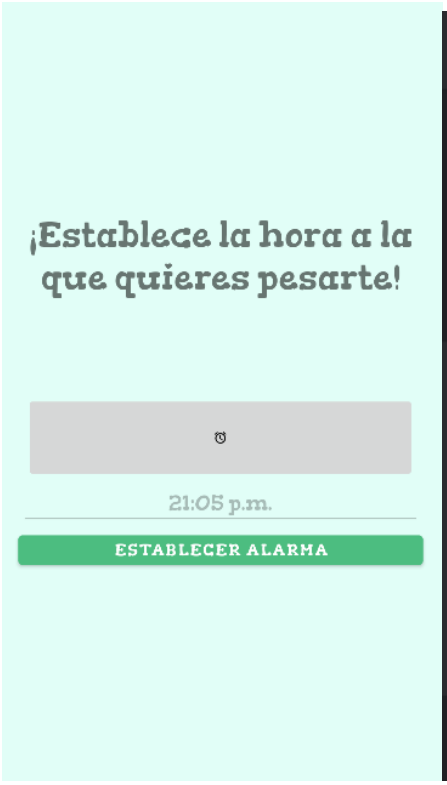


Ilustración 75 :Establecer alarma 1 *Ilustración 76 :Establecer alarma 2*



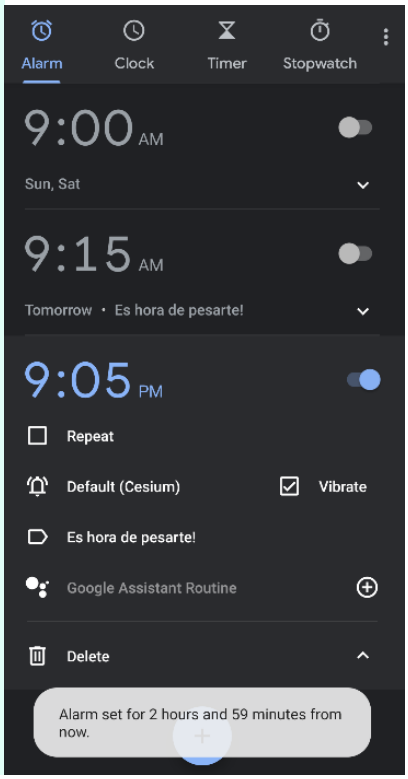


Ilustración 77 :Establecer alarma 3 *Ilustración 78 :Establecer alarma 4*

Actualizar peso

EL paciente puede actualizar su peso introduciendo el peso y después pulsando en "Actualizar"

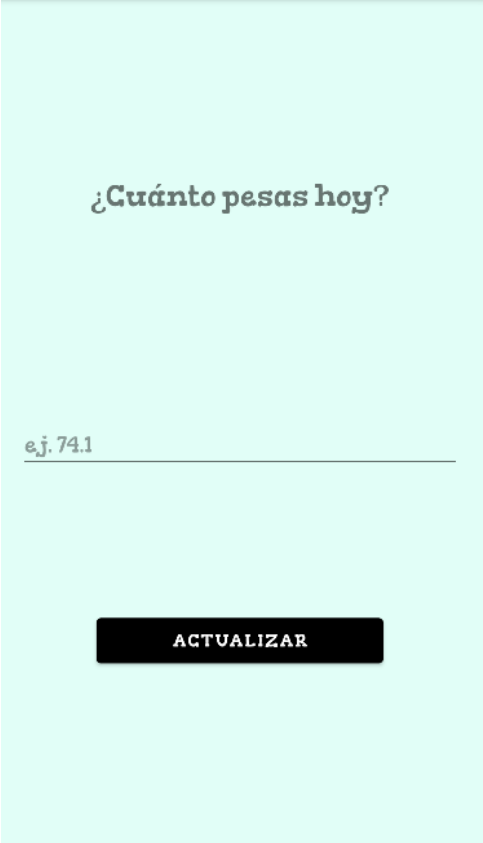


Ilustración 79: Actualizar peso Android

Seguimiento de dieta

Cada día el paciente tiene la posibilidad de realizar un seguimiento diario de su dieta para que su dietista asignado pueda verlo. Además, puede introducir un comentario que explique su día y puede además subir fotos para plasmar la alimentación que ha llevado.

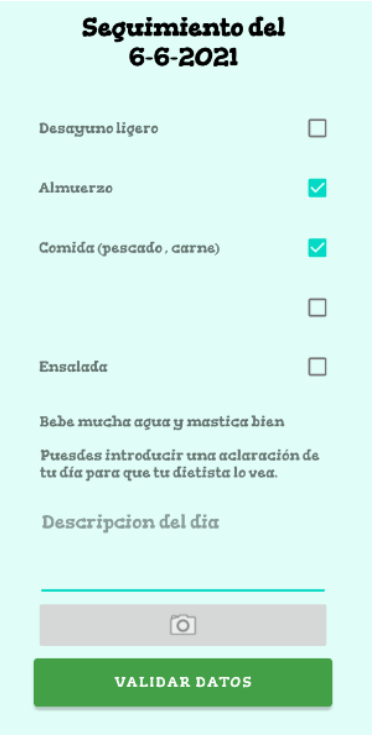


Ilustración 80: Seguimiento diario

Consejos para comer bien

El paciente tiene siempre a su disposición una serie de imágenes extraídas de EUFIC, una asociación sin ánimo de lucro orientada al consumidor.

Ilustración 81: Guía de comer sano por UEFIC

Ajustes de usuario

El paciente puede modificar todos los campos que se muestran en la imagen inferior y si pulsa en “MODIFICAR IMAGEN DE PERFIL” se abre la galería donde puede elegir una foto nueva de perfil.

Ilustración 82:Modificar perfil Android

Gestión de la dieta y el dietista.

Esta es la ventana donde el paciente puede ver los detalles de su dieta, sus seguimientos realizados y además acceder a la información de su dietista.

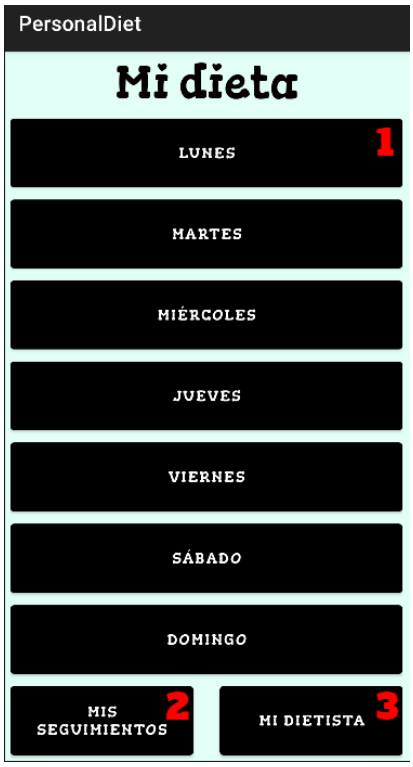


Ilustración 83: Mi dieta Android

1. Detalles de tu dieta: Aquí el paciente puede encontrar toda la información sobre el día que haya seleccionado, todas las comidas que tiene que realizar o no y algunos comentarios por parte del dietista si lo ve pertinente.

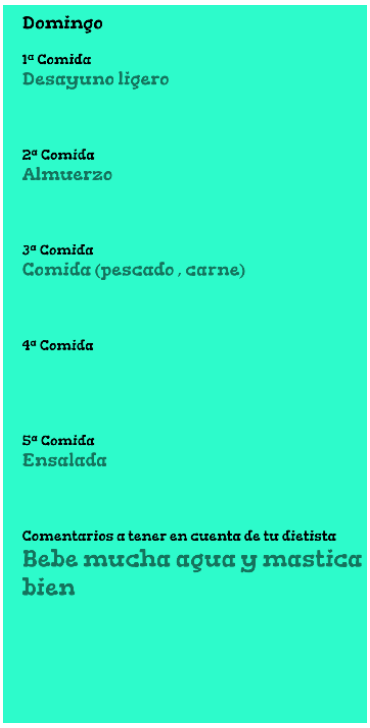
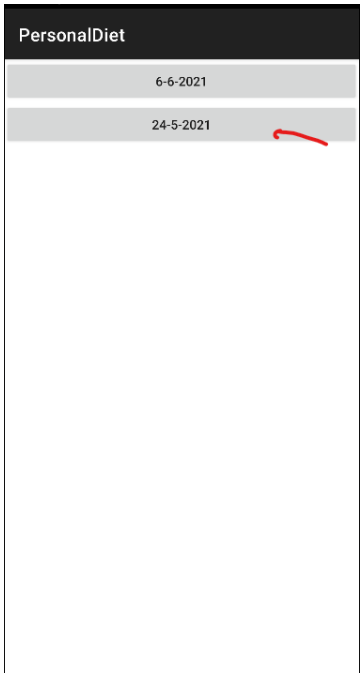


Ilustración 84:Detalles dieta

2. Mis seguimientos: el paciente puede visualizar todos los días que ha realizado el seguimiento. Pulsando en el día, se muestra todo lo que paciente indico ese día, si realizó bien la dieta o no , el peso que marcaba.



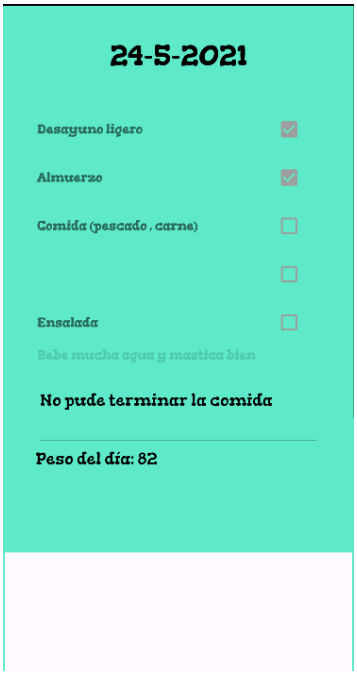


Ilustración 85: Lista de seguimientos Android *Ilustración 86: Seguimiento detalle*

3. Mi dietista: El paciente puede visualizar toda la información relacionada con el dietista que tiene asignado.

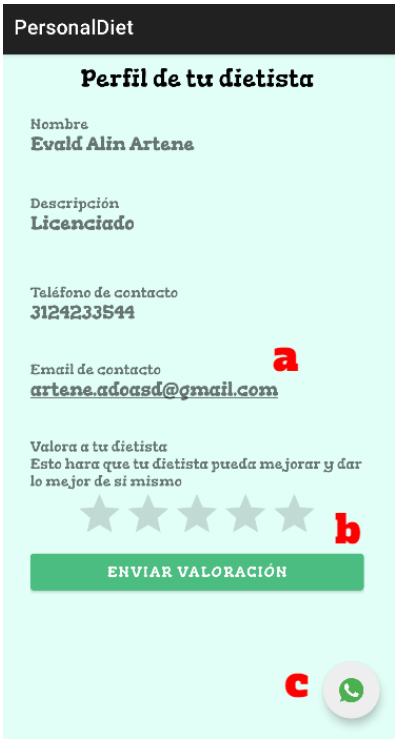


Ilustración 87: Perfil Dietista

a. Email: Pulsando en la dirección de correo se despliega la aplicación de correo electrónico que desee el paciente
b. Valoración del dietista: El paciente tiene la posibilidad de valorar a su dietista con la puntuación que desee para que aportar al resto de la comunidad su experiencia con ese dietista.
c. Integración con WhatsApp: El paciente puede abrir una conversación directa con el número de teléfono que ofrece el dietista a través de la aplicación de WhatsApp

Aplicación Web

Página principal, registro e inicio de sesión

Cuando se accede a la aplicación web, la primera vista que se ve es la siguiente:

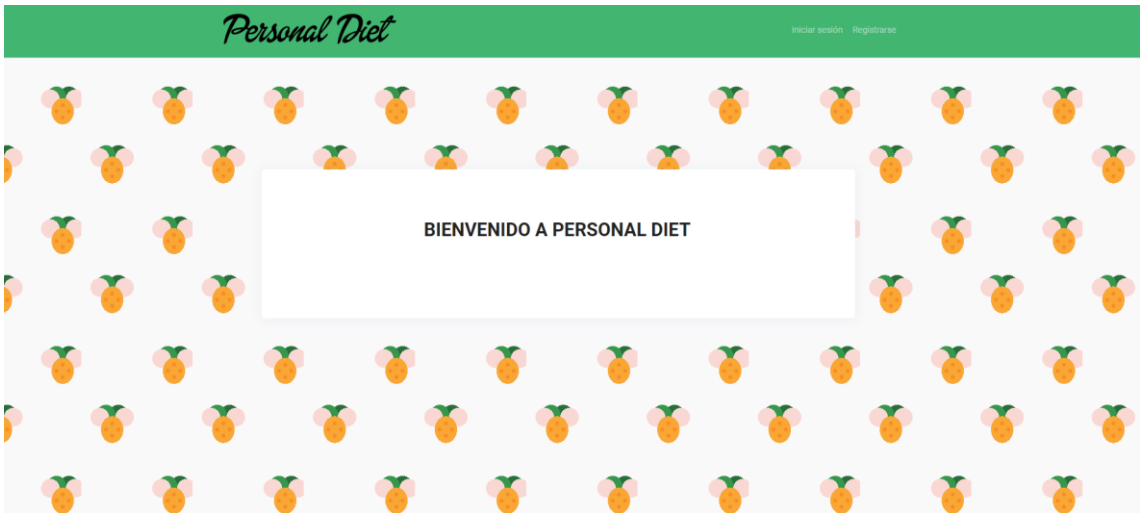


Ilustración 88 Web: Vista principal

Desde esta vista, se puede pulsar sobre el logo de ***PersonalDiet,*** lo que redirige al usuario a esta vista, pero en función de si se tiene iniciada sesión o no, y de con que usuario, administrador o dietista, las barras de navegación serán distintas. En este punto, se puede pulsar sobre el botón de iniciar sesión, en caso de ya tener una cuenta hecha, o en el botón de registrarse, para crear una nueva cuenta como dietista.

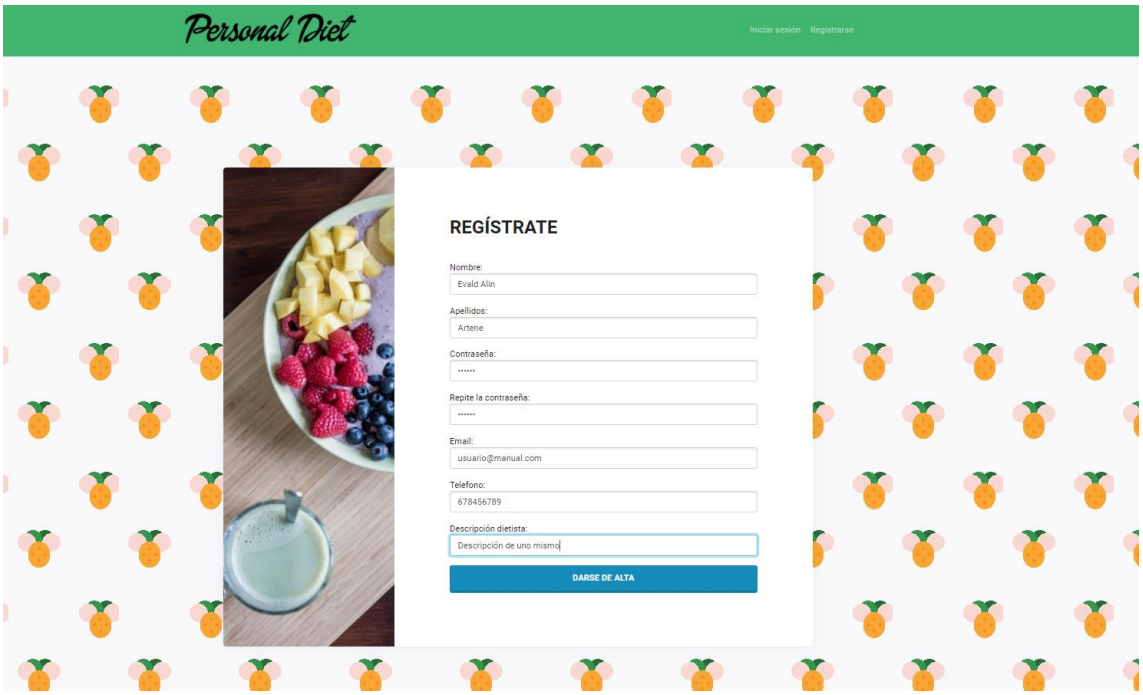


Ilustración 89 Web: Vista registro

En el caso de registrarse, aparece un pequeño formulario en el que se debe rellenar todos los campos que ahí aparecen. Una vez hecho, y completado de manera satisfactoria, se inicia sesión en el sistema como un nuevo usuario con las credenciales elegidas por uno mismo.

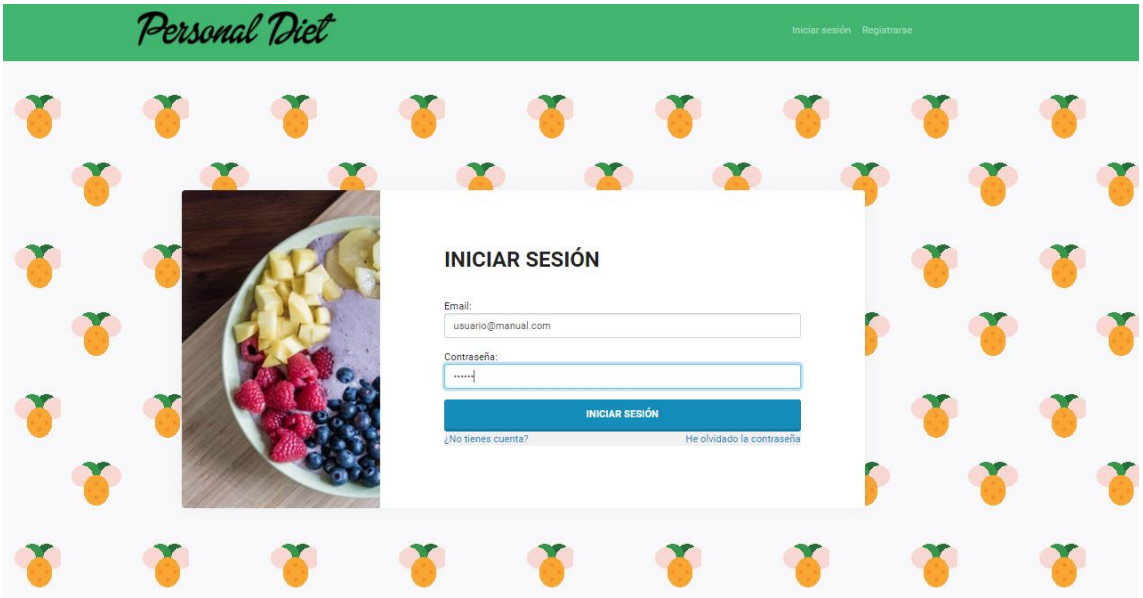


Ilustración 90 Web: Vista inicio sesión

Como se ha dicho anteriormente, al iniciar sesión como dietista, la página web a la que se redirige es la de la pagina principal de los dietistas.

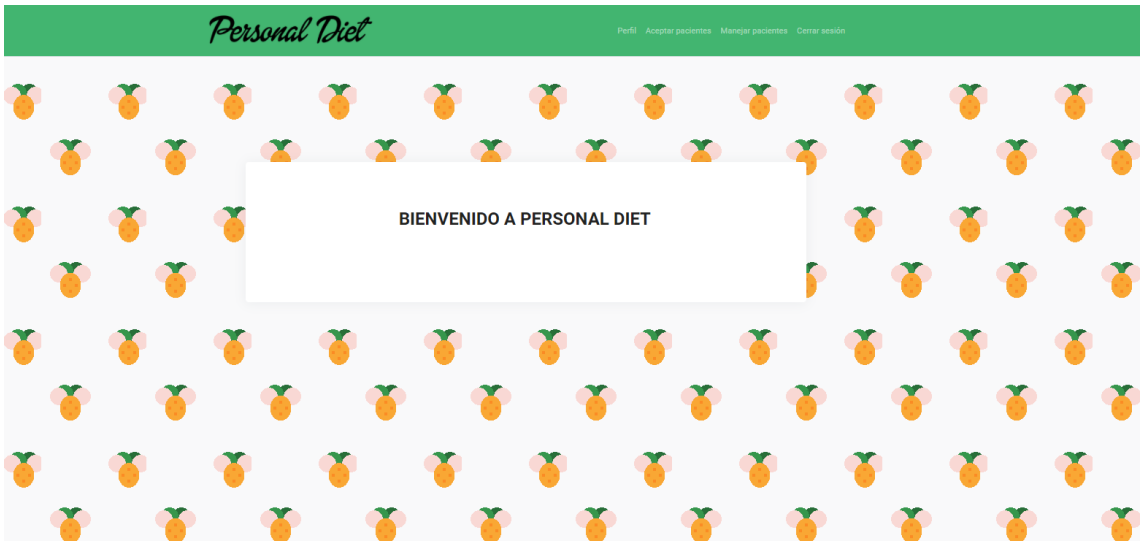


Ilustración 91 Web: Vista principal dietista

Como se observa, la barra de navegación superior es distinta a la de un usuario que no ha iniciado sesión todavía en la aplicación. A su vez, la diferencia que se tiene al iniciar como administrador es que se muestra otra barra de navegación, y es la siguiente:

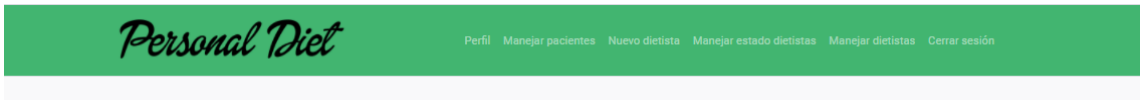


Ilustración 92 Web: Vista barra administrador

Funcionalidades dietista

Una vez iniciada sesión como dietista, se puede acceder a los distintos apartados que este tiene. Cuando se registra un dietista en la aplicación web, y todavía no es admitido por el administrador, la única función que puede realizar el dietista es la de ver su perfil personal, y las acciones que en dicha vista se enseñan.

A continuación, se muestra el perfil del dietista, junto con la vista que este recibe si trata de acceder a los apartados de *"Aceptar pacientes"* y "*Manejar pacientes"*.

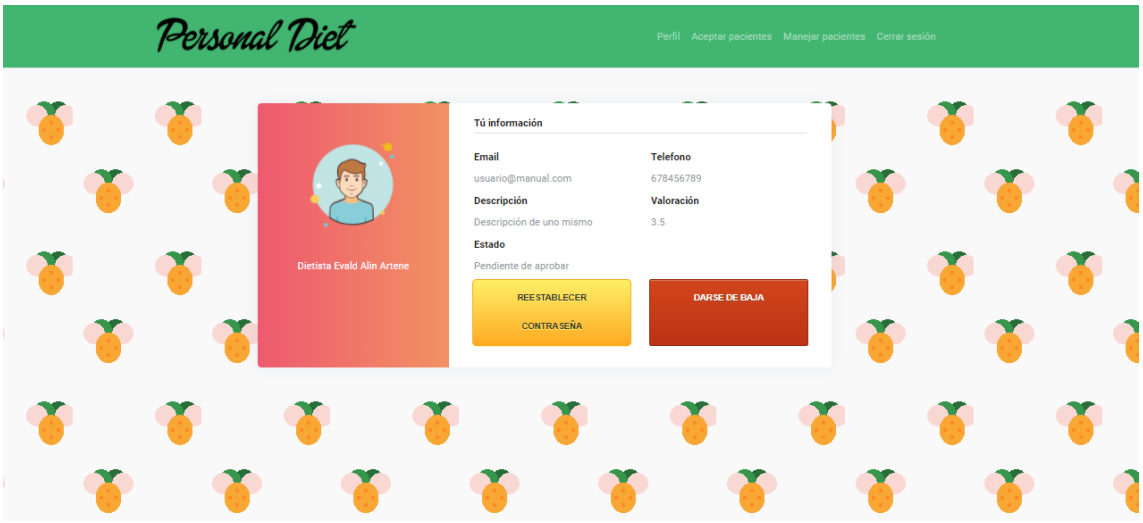


Ilustración 93 Web: Vista perfil dietista

Dietista con estado "Pendiente de aprobar"

Ilustración 94 Web: Vista dietista no aprobado

Dietista con estado "Aprobado"

Una vez aceptado en la aplicación por el administrador, el dietista puede acceder a las demás vistas, y, por tanto, funcionalidades, de la aplicación web.

En primer lugar, si se accede a "*Aceptar pacientes",* en esta pestaña se muestra las solicitudes que este tiene de pacientes que quieren los servicios del dietista en cuestión.

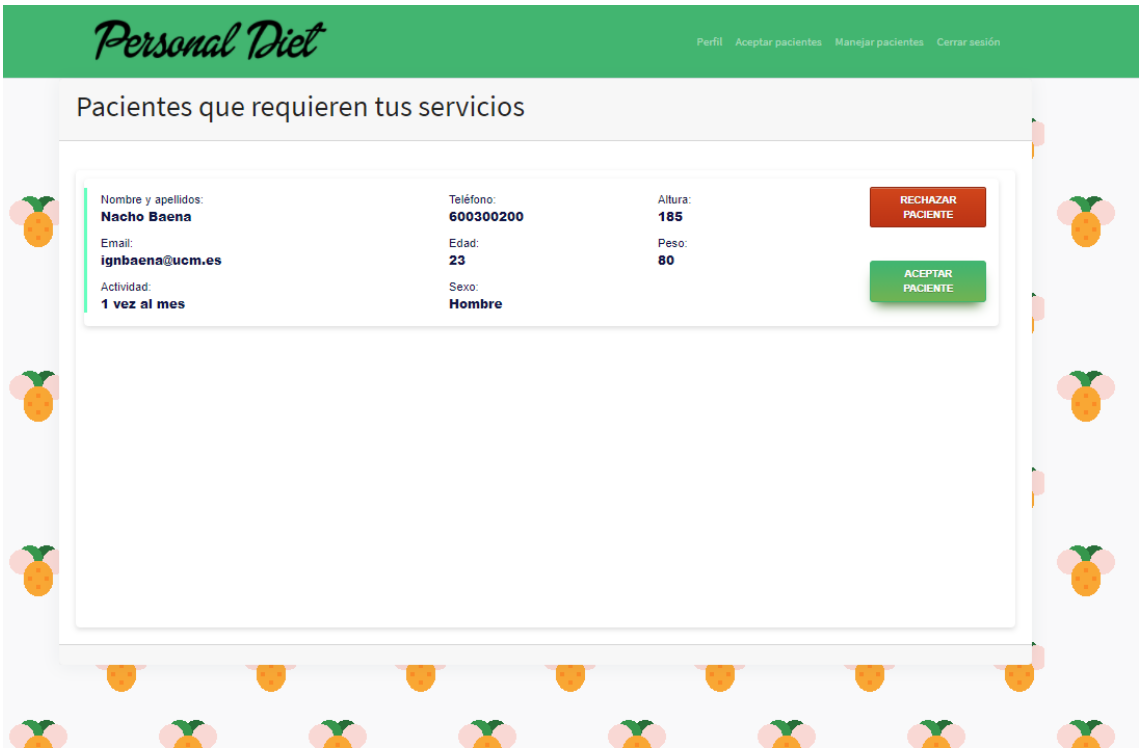


Ilustración 95 Web: Vista aceptar pacientes

Esta es la visual que se recibe. Si se pulsa sobre el botón de "*Rechazar paciente",* se vuelve a cargar dicha página con las restantes solicitudes, y se le notifica al paciente de que ha sido rechazado y debe elegir nuevamente un dietista. Si se pulsa sobre el botón "*Aceptar paciente",* se asignará el paciente al dietista, y se procederá a realizar la dieta al paciente mediante el siguiente formulario.

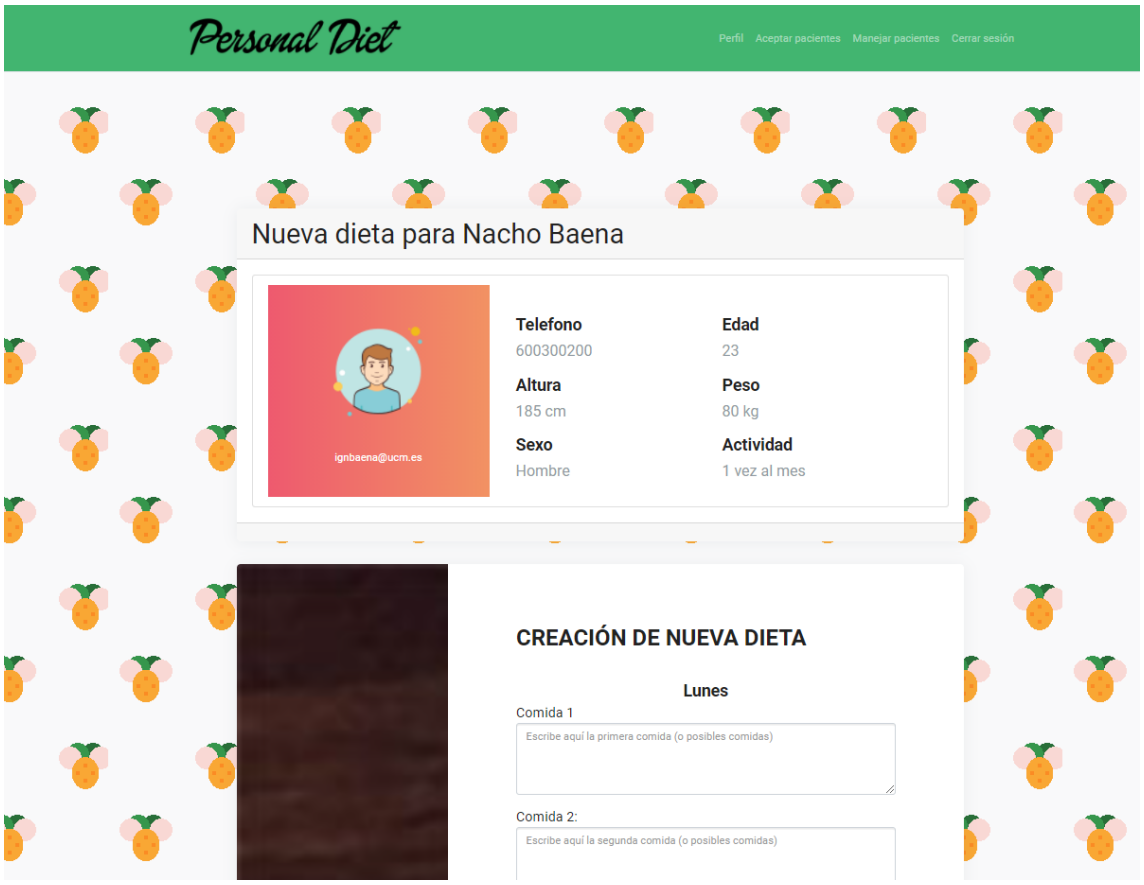


Ilustración 96 Web: Vista nueva dieta

En esta vista, se muestra la información que el paciente ha registrado. El formulario que se presenta es muy extenso, pues el dietista debe crear, para cada uno de los días que tiene la semana, al menos, dos comidas, junto con una descripción para cada uno de los días. Si el dietista no rellena las dos primeras comidas de cada día, así como el comentario sobre cada día, no podrá avanzar de vista.

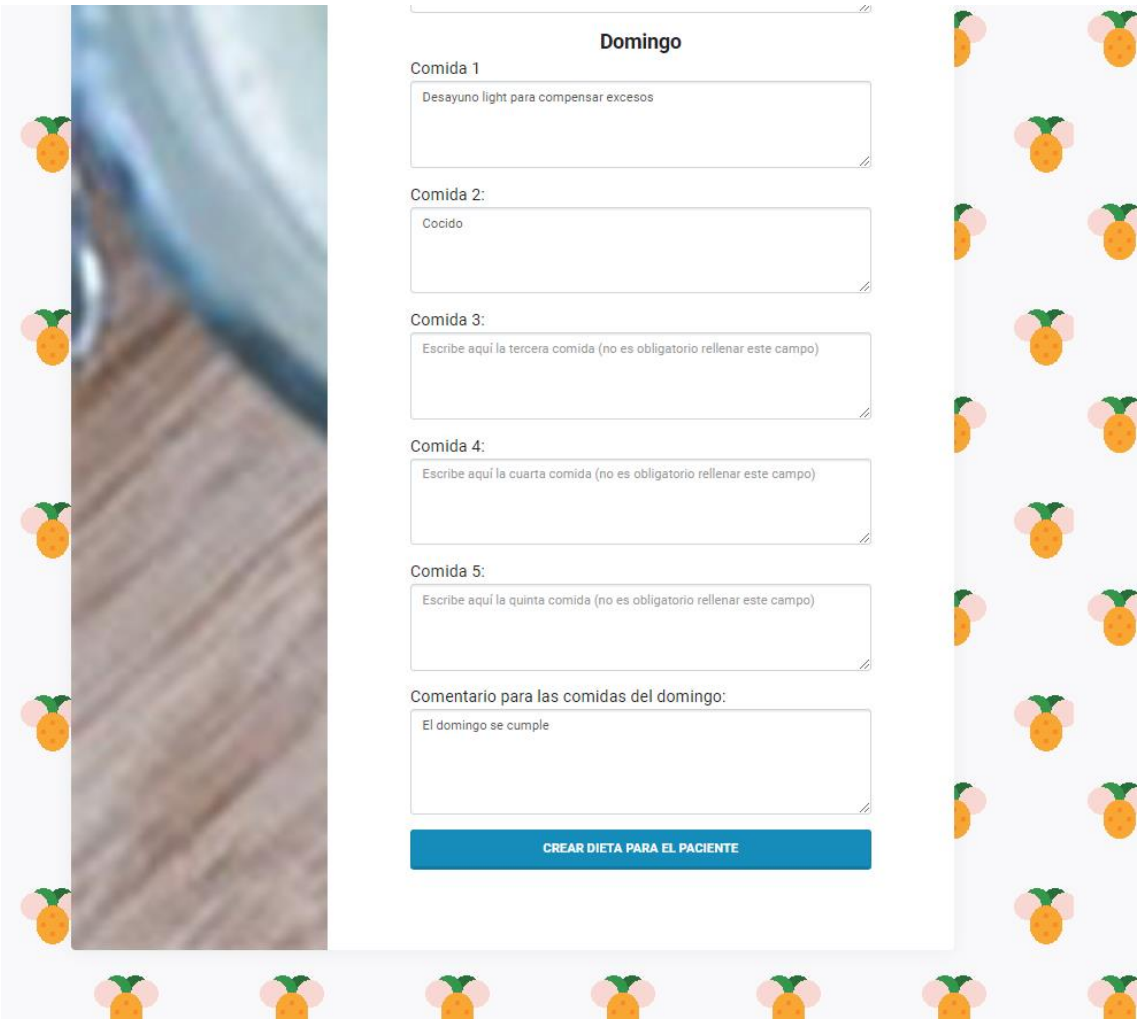


Ilustración 97 Web: Vista nueva dieta 2

Cuando se pulsa el botón de realizar dieta, se crea esta y se redirige al dietista a la página principal del paciente, que es la siguiente:

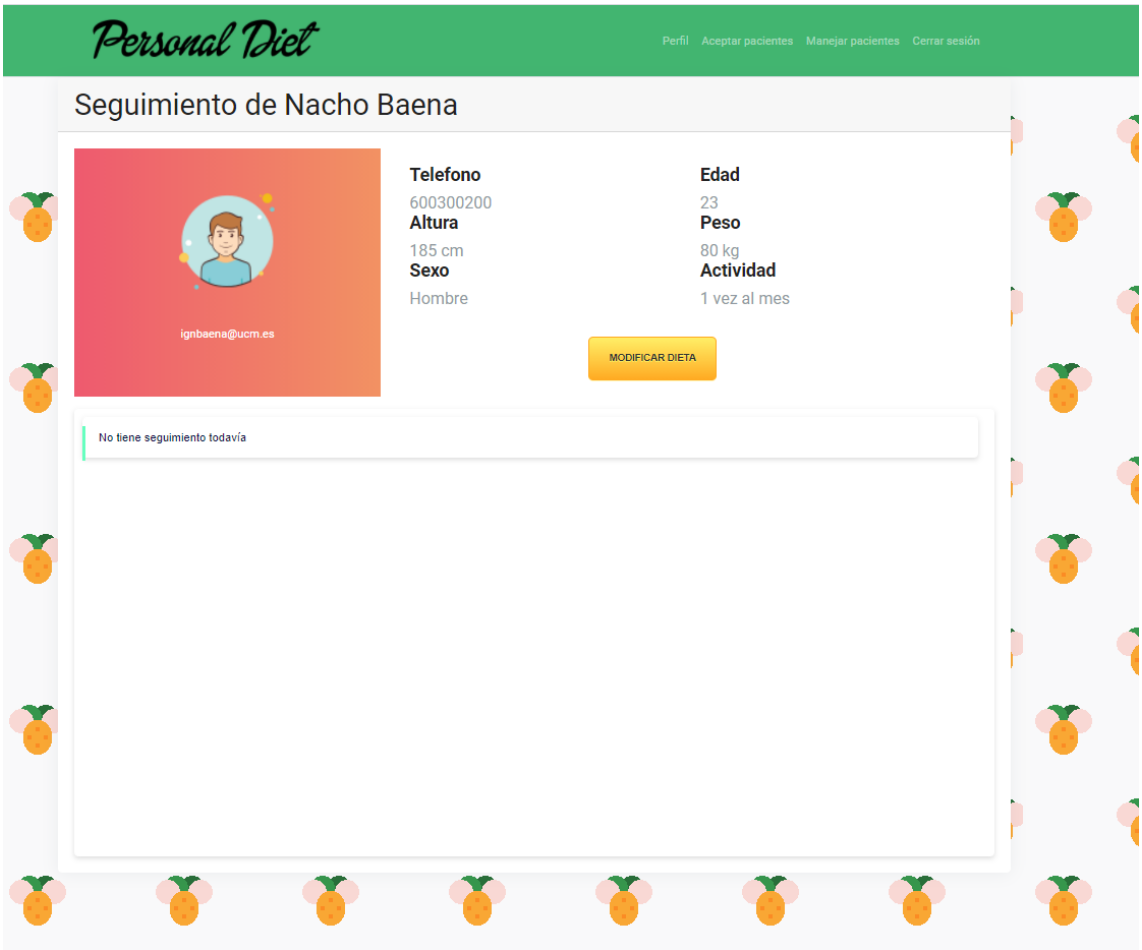


Ilustración 98 Web: Vista seguimiento

En esta vista, se puede observar nuevamente la información de perfil del usuario, así como un botón para modificar la dieta que se le ha creado, y los seguimientos que tiene el paciente. Si no tiene seguimientos, como es el caso, se notifica de ello.

Si se pulsa en el botón de modificar dieta, se redirige un formulario muy parecido al de creación de dietas, con la diferencia en que los campos de este están rellenos con los valores de la dieta en cuestión.

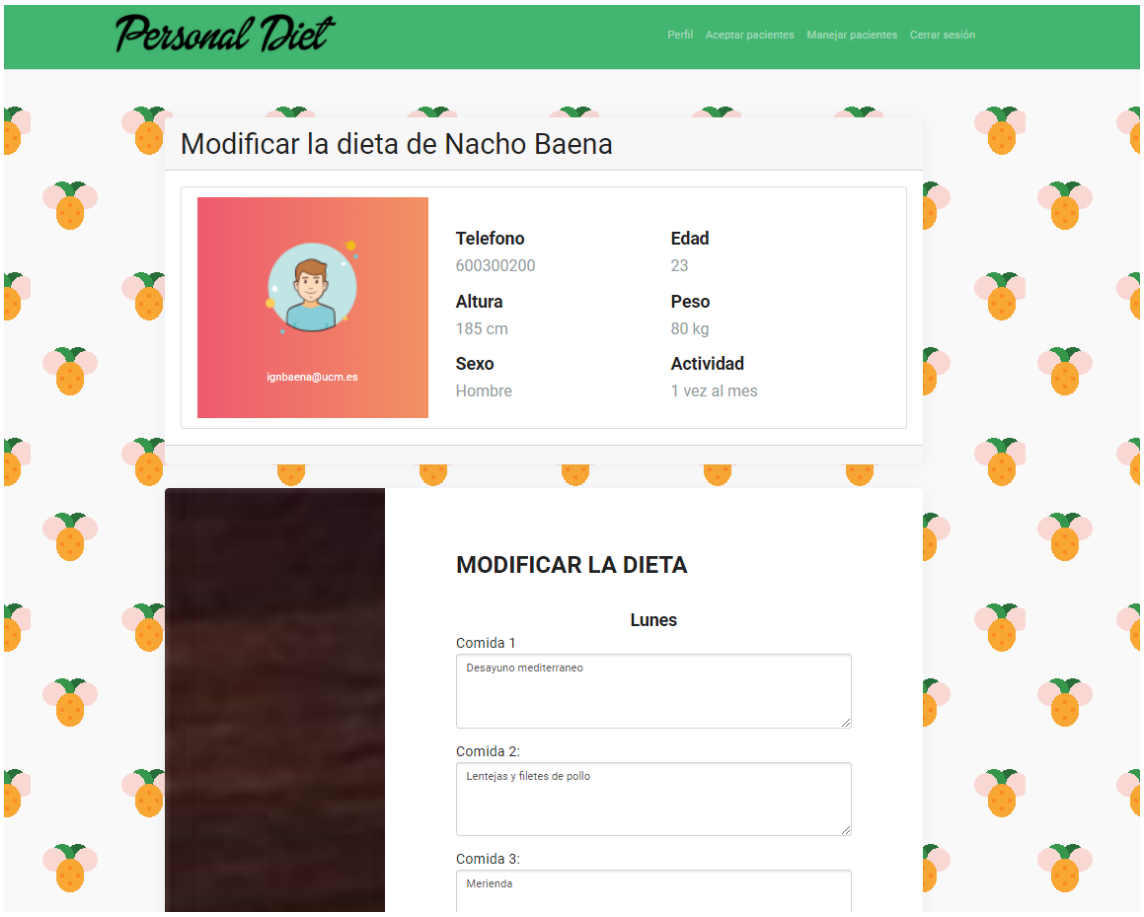


Ilustración 99 Web: Vista modificar dieta

Si se pulsa sobre la pestaña de “Manejar pacientes”, aparece la siguiente visual.

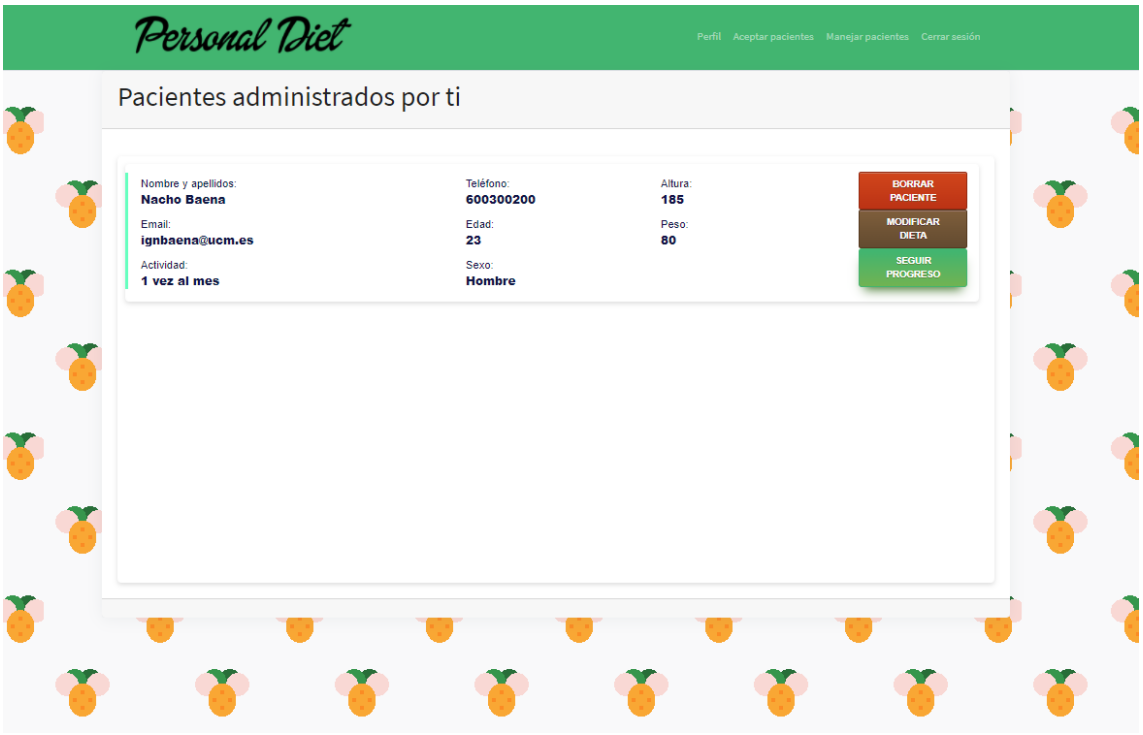


Ilustración 100 Web: Vista manejar pacientes dietista

En esta vista, se pueden encontrar todos los pacientes que el dietista tiene ya asignados, así como varios botones que realizan diversas acciones.
En caso de pulsar sobre el botón de *"Borrar paciente",* se elimina al paciente de la lista de pacientes que el dietista tiene asignado, así como la dieta que este tenía, y se notifica al paciente de esta acción.
En caso de pulsar sobre el botón de *"Modificar dieta",* se redirige al formulario arriba mostrado de modificación de dieta. En el caso de que, al aceptar el paciente, no se crea la dieta en ese momento, en lugar de dicho botón, se muestra otro, "Crear dieta", que redirige al formulario de creación de dietas.
En caso de pulsar sobre el botón de *"Seguir progreso",* se redirige a la vista en el que se muestra el progreso del paciente.

Funcionalidades Administrador

En el caso de iniciar sesión como administrador, como se mencionado anteriormente, la barra de navegación, y, por tanto, las funcionalidades a las que se pueden acceder y realizar, son distintas.
En primer lugar, el perfil del administrador cambia, pues no se deja que el administrador pueda darse de baja, ya que es una pieza fundamental en el desarrollo del correcto funcionamiento de la aplicación.

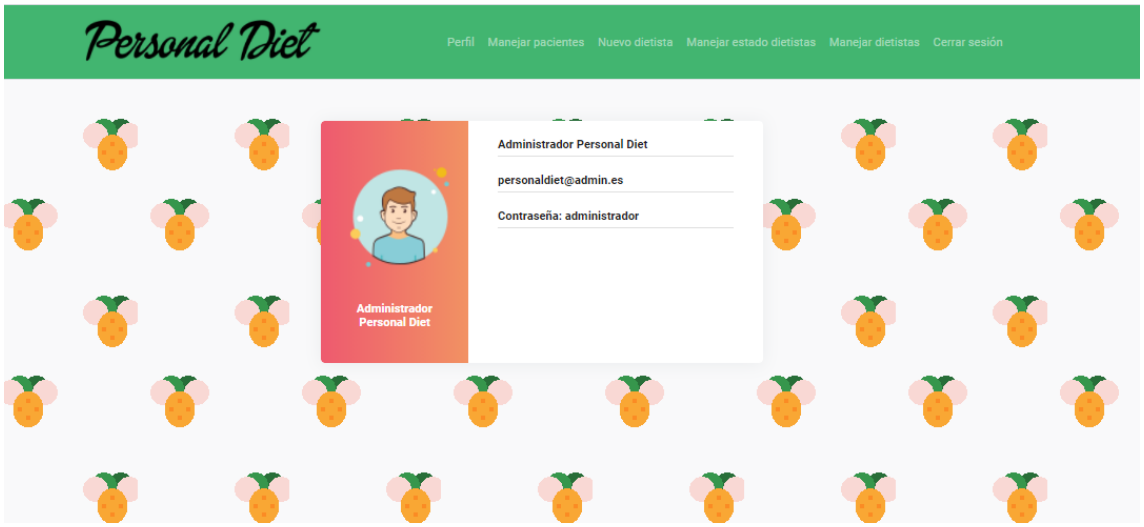


Ilustración 101 Web: Vista perfil administrador

La información que aquí se muestra es meramente informativa.

En segundo lugar, si el administrador pulsa sobre el botón *"Manejar pacientes"* de la barra de navegación, se redirige la siguiente visual.

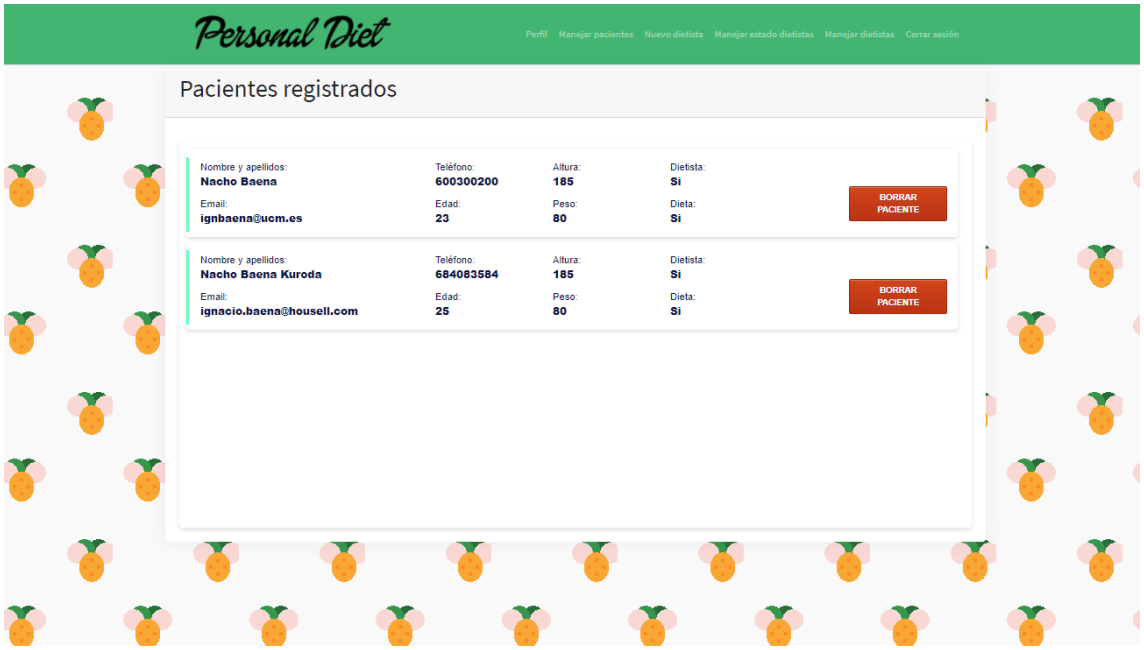


Ilustración 102 Web: Vista manejar pacientes administrador

En esta vista, aparecen todos los pacientes que se encuentran dados de alta en la aplicación, así como cierta información pertinente. En el caso de pulsar sobre el botón de *"Borrar paciente"* se elimina al paciente de la base de datos, así como la dieta que tiene, y también de la lista de pacientes del dietista.

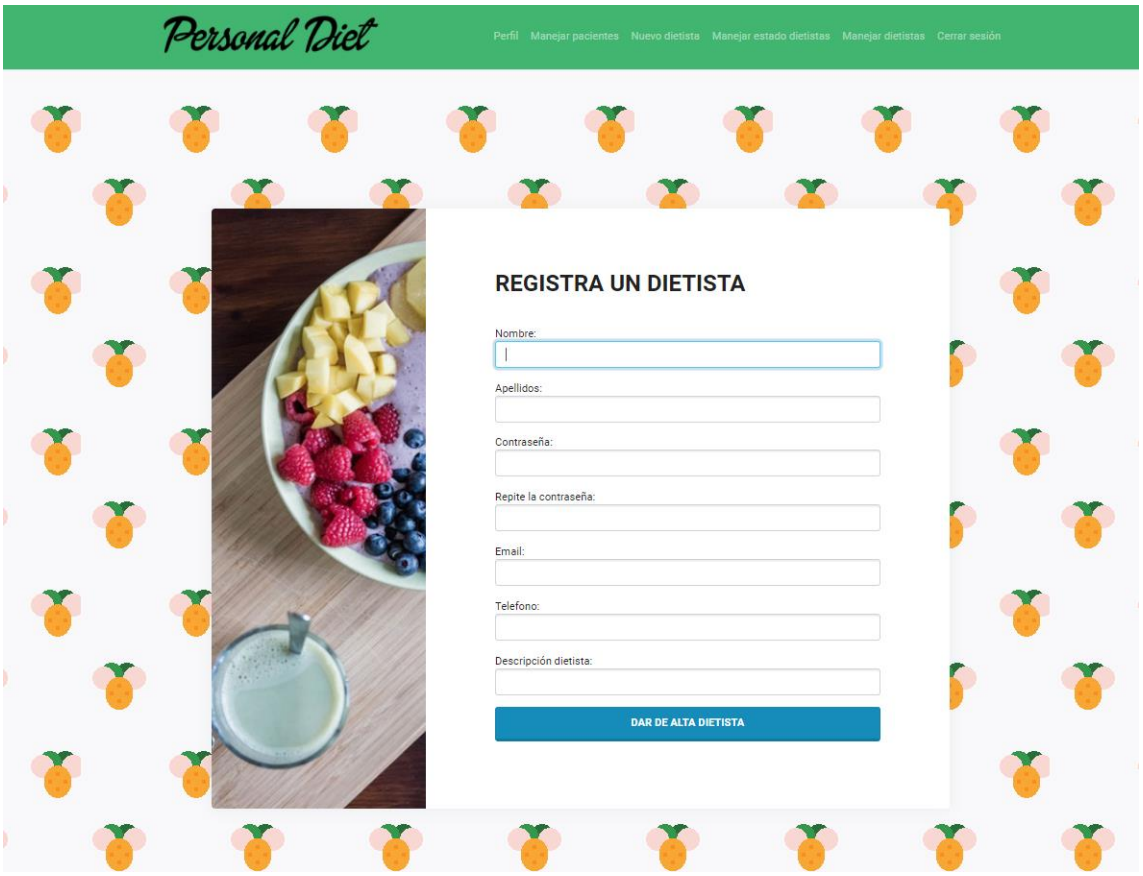


Ilustración 103 Web: Vista crear dietista

A continuación, si se pulsa sobre el botón de *"Nuevo dietista"*, se redirige a un formulario de creación de usuario en la aplicación web igual que el que tienen los dietistas para registrarse, con la única diferencia de que los dietistas creados desde esta visual aparecen con estado *"Aprobado"* directamente, sin que haga falta que el administrador les acepte posteriormente.

Si se pulsa sobre el botón de *"Manejar estado dietistas",* se redirige a la siguiente vista.

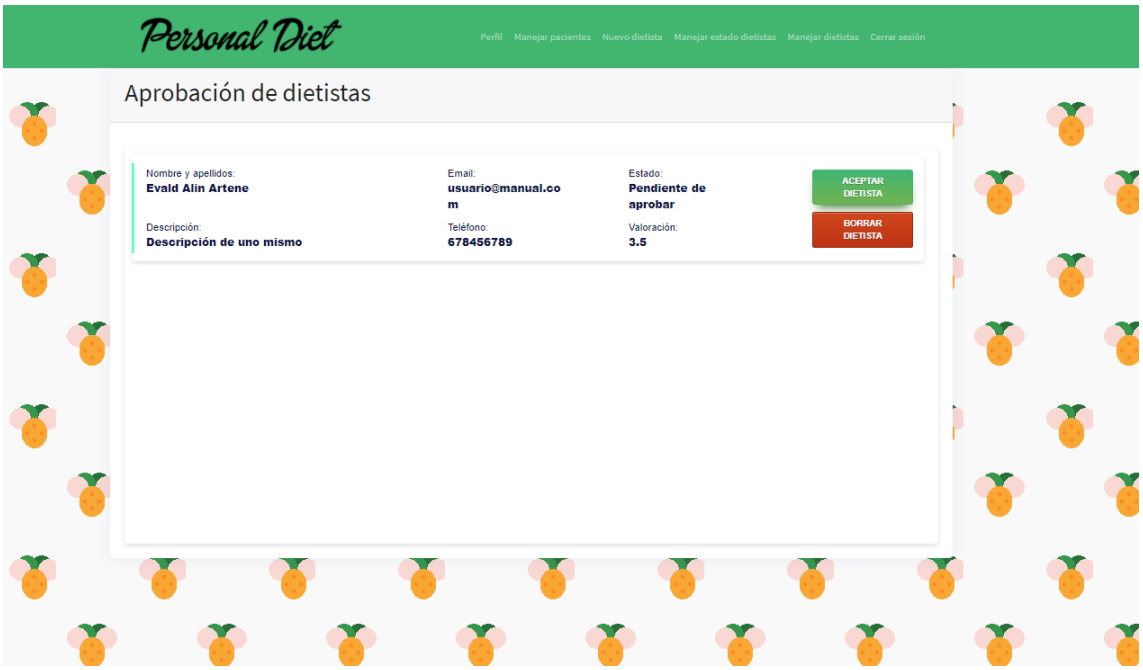


Ilustración 104 Web: Vista estado dietistas

En esta vista del administrador, aparecen los usuarios que se han dado de alta en la aplicación web por cuenta propia, por lo que el administrador les puede aceptar pulsando sobre el botón *"Aceptar dietista",* lo que cambia su estado en la base de datos a *"Aprobado",* o rechazarles, y por tanto borrarles de la aplicación, pulsando sobre *"Borrar dietista"*.

Finalizando, si se pulsa sobre el botón *"Manejar dietistas"* de la barra de navegación del administrador, se redirige a la siguiente visual.

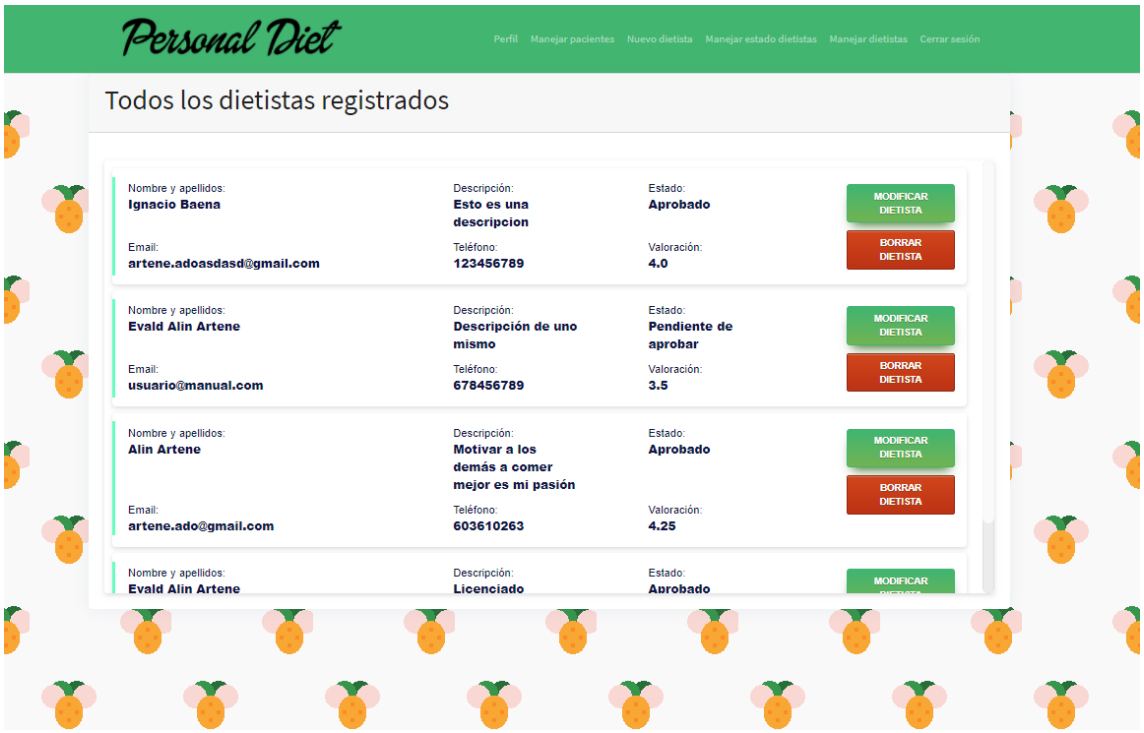


Ilustración 105 Web: Vista manejar dietistas

Esta vista muestra todos los dietistas que se encuentran datos de alta en la aplicación, cualquiera que sea su estado, y desde esta visual se puede modificar cierta información del dietista, pulsando sobre el botón *"Modificar dietista"*, o borrar el dietista de la aplicación, pulsando el otro botón, lo que provocará que se eliminen todas las dietas que este tiene realizadas a los distintos pacientes que tiene a su cargo notificando a cada uno de ellos de esto.

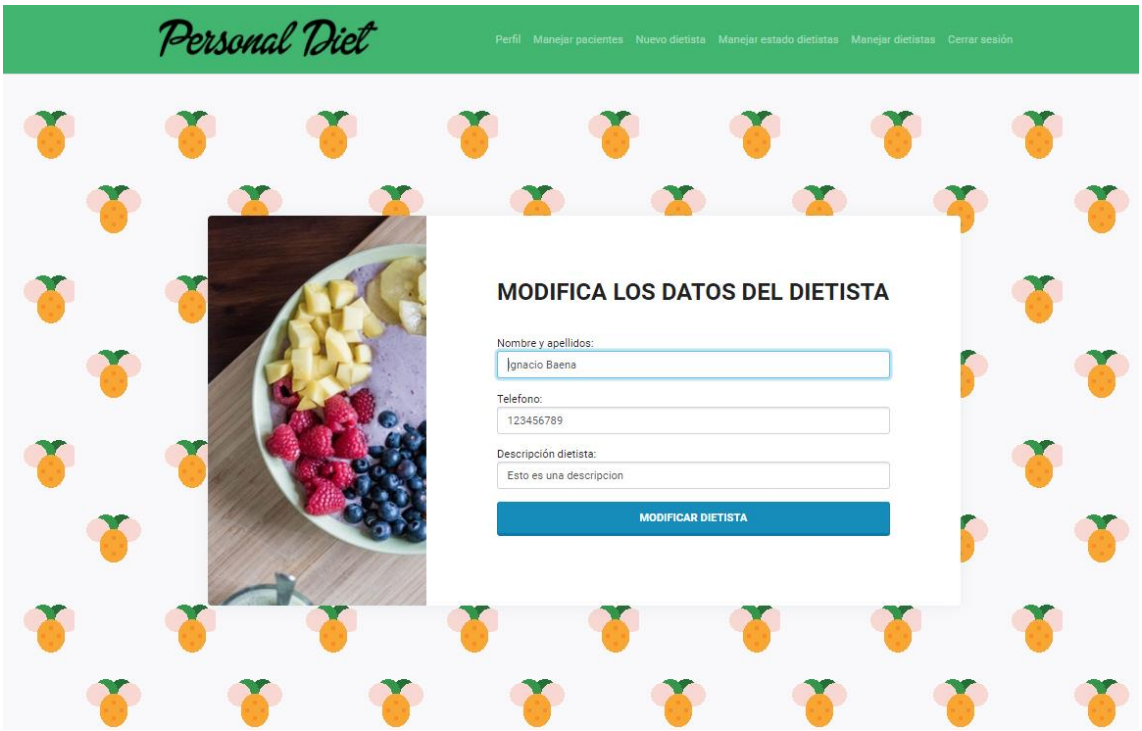


Ilustración 106Web: Vista modificar dietista